Ana Belén Cambra Linés

Scene understanding for interactive applications

Departamento
Informática e Ingeniería de Sistemas

Director/es
Muñoz Orbañanos, Adolfo
Murillo Arnal, Ana Cristina

ISSN 2254-7606

© Universidad de Zaragoza
Servicio de Publicaciones

ISSN 2254-7606

Tesis Doctoral

SCENE UNDERSTANDING FOR INTERACTIVE APPLICATIONS

Autor

Ana Belén Cambra Linés

Director/es

Muñoz Orbañanos, Adolfo
Murillo Arnal, Ana Cristina

UNIVERSIDAD DE ZARAGOZA
Informática e Ingeniería de Sistemas

2018

Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Scene understanding for interactive applications

Ana Belén Cambra Linés

Ph.D. Dissertation

Supervisors: Adolfo Muñoz Orbañanos and Ana Cristina Murillo Arnal

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza
October 2017

Resumen

Para interactuar con el entorno, es necesario entender que está ocurriendo en la escena donde se desarrolla la acción. Décadas de investigación en el campo de la vision por computador han contribuido a conseguir sistemas que permiten interpretar de manera automática el contenido en una escena a partir de información visual. Se podría decir el objetivo principal de estos sistemas es replicar la capacidad humana para extraer toda la información a partir solo de datos visuales. Por ejemplo, uno de sus objetivos es entender como percibimos el mundo en tres dimensiones o como podemos reconocer sitios y objetos a pesar de la gran variación en su apariencia. Una de las tareas básicas para entender una escena es asignar un significado semántico a cada elemento (píxel) de una imagen. Esta tarea se puede formular como un problema de etiquetado denso el cual especifica valores (etiquetas) a cada pixel o region de una imagen. Dependiendo de la aplicación, estas etiquetas pueden representar conceptos muy diferentes, desde magnitudes físicas como la información de profundidad, hasta información semántica, como la categoría de un objeto. El objetivo general en esta tesis es investigar y desarrollar nuevas técnicas para incorporar automáticamente una retroalimentación por parte del usuario, o un conocimiento previo en sistemas inteligente para conseguir analizar automáticamente el contenido de una escena. en particular, esta tesis explora dos fuentes comunes de información previa proporcionado por los usuario: interacción humana y etiquetado manual de datos de ejemplo.

La primera parte de esta tesis esta dedicada a aprendizaje de información de una escena a partir de información proporcionada de manera interactiva por un usuario. Las soluciones que involucran a un usuario imponen limitaciones en el rendimiento, ya que la respuesta que se le da al usuario debe obtenerse en un tiempo interactivo. Esta tesis presenta un paradigma eficiente que aproxima cualquier magnitud por pixel a partir de unos pocos trazos del usuario. Este sistema propaga los escasos datos de entrada proporcionados por el usuario a cada pixel de la imagen. El paradigma propuesto se ha validado a través de tres aplicaciones interactivas para editar imágenes, la cuales requieren un conocimiento por pixel de una cierta magnitud, con el objetivo de simular distintos efectos.

Otra estrategia común para aprender a partir de información de usuarios es diseñar sistemas supervisados de aprendizaje automático. En los últimos años, las redes neuronales convolucionales han superado el estado del arte de gran variedad de problemas de reconocimiento visual. Sin embargo, para nuevas tareas, los datos necesarios de entrenamiento pueden no estar disponibles y recopilar suficientes no es siempre posible. La segunda parte de esta tesis explora como mejorar los sistema que aprenden etiquetado denso semántico a partir de imágenes previamente etiquetadas por los usuarios. En particular, se presenta y validan estrategias, basadas en los dos principales enfoques para transferir modelos basados en deep learning, para segmentación semántica, con el objetivo de poder aprender nuevas clases cuando los datos de entrenamiento no son suficientes en cantidad o precisión. Estas estrategias se han validado en varios entornos realistas muy diferentes, incluyendo entornos urbanos, imágenes aereas y imágenes submarinas.

Abstract

In order to *interact with the environment*, it is necessary to understand what is happening on it, on the *scene* where the action is ocurring. Decades of research in the computer vision field have contributed towards automatically achieving this *scene understanding* from visual information. Scene understanding is a very broad area of research within the computer vision field. We could say that it tries to replicate the human capability of extracting plenty of information from visual data. For example, we would like to understand how the people perceive the world in three dimensions or can quickly recognize places or objects despite substantial appearance variation.

One of the basic tasks in scene understanding from visual data is to assign a semantic meaning to every element of the image, i.e., assign a concept or object label to every pixel in the image. This problem can be formulated as a dense image labeling problem which assigns specific values (labels) to each pixel or region in the image. Depending on the application, the labels can represent very different concepts, from a physical magnitude, such as depth information, to high level semantic information, such as an object category. The general goal in this thesis is to investigate and develop new ways to automatically incorporate human feedback or prior knowledge in intelligent systems that require scene understanding capabilities. In particular, this thesis explores two common sources of prior information from users: human interactions and human labeling of sample data.

The first part of this thesis is focused on learning complex scene information from interactive human knowledge. Interactive user solutions impose limitations on the performance where the feedback to the user must be at interactive rates. This thesis presents an efficient interaction paradigm that approximates any per-pixel magnitude from a few user strokes. It propagates the sparse user input to each pixel of the image. We demonstrate the suitability of the proposed paradigm through three interactive image editing applications which require per-pixel knowledge of certain magnitude: simulate the effect of depth of field, dehazing and HDR tone mapping.

Other common strategy to learn from user prior knowledge is to design supervised machine-learning approaches. In the last years, Convolutional Neural Networks (CNNs) have pushed the state-of-the-art on a broad variety of visual recognition problems. However, for new tasks, enough training data is not always available and therefore, training from scratch is not always feasible. The second part of this thesis investigates how to improve systems that learn dense semantic labeling of images from user labeled examples. In particular, we present and validate strategies, based on common transfer learning approaches, for semantic segmentation. The goal of these strategies is to learn new specific classes when there is not enough labeled data to train from scratch. We evaluate these strategies across different environments, such as autonomous driving scenes, aerial images or underwater ones.

Contents

Chapter 1

Introduction

The computer vision field is receiving increasing interest from plenty of emerging technologies, such as autonomous driving or augmented reality, as well as more mature technologies, such as social networks or automated surveillance systems, partly thanks to recent advances both in the related hardware technologies and algorithmic solutions. One of the most challenging topics in the field is to achieve automatic scene understanding. In other words, to interact with the environment, with the scenarios where the actions and activities are happening, it is necessary to understand what components are placed on those scenes, as well as what actions or events are happening there.

For humans, the sense of sight is essential and is one of the most complex systems in our body. While humans use their eyes and brain to sense the world around them, computer vision is the scientific field that aims to enable machines with a similar capability. Decades of research in computer vision field have already shown great achievements on enabling intelligent systems with high-level semantic understanding of the content from digital images or videos. Nowadays, we can automate some tasks that the human visual system can do, such as detecting and recognizing faces, with impressive levels of accuracy, reaching human levels [1]. However, human perception can perform tasks that automated systems are still far from achieving, such as recognizing and locating all objects in an image [2] or predicting the trajectory of a pedestrian based on the previous seen positions [3].

Furthermore, not only humans can still recognize and understand more information from visual data than machines, but they also do it extremely efficiently (almost instantly). However, complex computer vision algorithms frequently need a longer response time and a large amount of resources to yield a solution. Therefore, an even more demanding challenge is not only to replicate the human capabilities for visual scene understanding, but to target also the time lapse that a person requires to do it. This efficiency and speed requirement would allow us to apply scene understanding algorithms for applications that actually interact with humans in real scenarios, and therefore have restrictions on the response time that is acceptable.

1.1 Motivation and Context

Scene understanding is an essential functionality of the human vision system and also a major goal of computer vision research. The goal of scene understanding is to obtain as much knowledge of the given visual data as possible.

We can consider object recognition and detection a relevant and well studied sub-topic of scene understanding. Classification and visual categorization of elements in a given image are some of the most investigated topics in computer vision. In the past, object recognition research has had a lot of focus on recognizing single objects [4], and understanding the scene as a whole [5]. More recently, research goals have shifted to more complex tasks, towards figuring out, for example, the relationship between multiple objects in a single image.

In this respect, towards the goal of analyzing all the parts or components of the image and their co-ocurrence, we can define the problem of dense image labeling, that we can consider another sub-topic of scene understanding. This problem aims to automatically label every pixel in the image with certain concept or category. The dense labeling task can target to make use from somehow physical properties, such as depth or 3D information, to higher level semantic concepts, such as recognizing objects or identifying materials. Approaches working towards this latter case are often referred to as semantic labeling or semantic parsing of images.

Plenty of technologies and applications have started to include computer vision components, thanks to a combination of multiple reasons, such as recent advances in computers, GPU, cloud connectivity, better and cheaper sensors... These advances could be somehow summarized in better performance of vision based automated tasks. Many of these applications, such as the examples shown in Fig. 1.1, actually require to perform different tasks related to automated scene understanding. Next we describe just a few of these application areas which benefit from automated visual scene understanding. The large variety of applications which benefit from advances on this topic emphasizes the significance of the problem studied and the impact of new contributions related to it and its applicability to real world use cases.

Image annotation. Automatically recognizing objects in the scene helps tasks such as automatic annotation of visual media content. This directly enables to automatically organize and index large amounts of data, in this case visual data. We find several web search engines, such as the *Search By Image* [1] search provided by Google, which let us perform a search providing an image as input. After you supply a photo, the system automatically analyzes the image to identify the content on it and return similar results.

3D modeling of the scene. Advanced image processing techniques rely on specific per-pixel information about certain magnitudes, for instance augmented reality applications

[1] https://images.google.com/

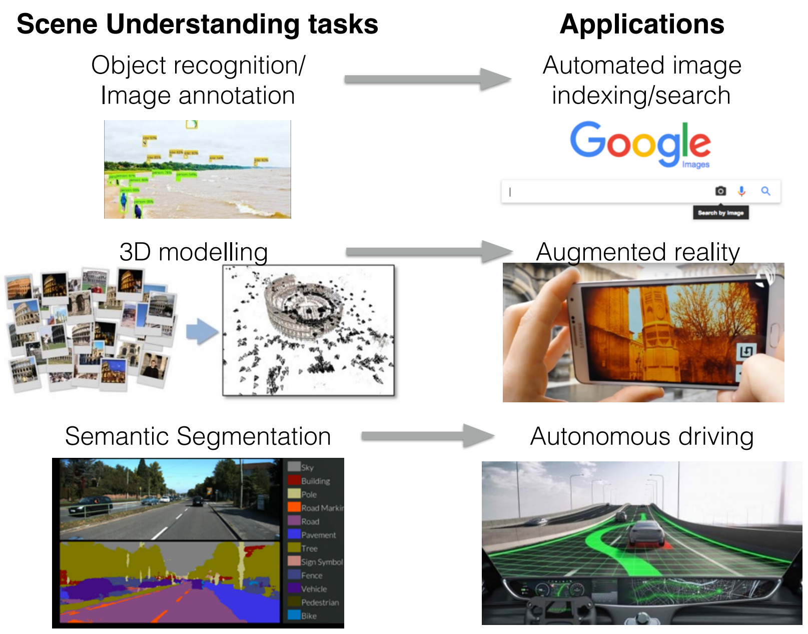


Figure 1.1: Sample applications that benefit from automatic scene understanding algorithms.

need depth information. Augmented reality allows us to insert virtual objects in an image or video, combining both real and virtual worlds. In order to accomplish this goal, it is important a good alignment between the real and virtual elements, which requires accurate knowledge of the 3D information of the scene elements. This information can be obtained applying different techniques to obtain a depth estimation in order to build the 3D model of the scene.

Semantic segmentation. Scene understanding is crucial for autonomous systems to operate in real-world scenarios. Applications such as autonomous driving have already shown to benefit from an initial semantic segmentation of the scene, which provides essential information such as drivable regions of the scene, or location of pedestrian or other obstacles. Visual based scene understanding is an important scientific problem that has to be addressed in order to provide the information about the environment needed to be able to automate numerous tasks in different application fields. This topic is one of the main problems studied in this thesis, as detailed in the following sections.

1.2 Scene understanding

Scene understanding is a very broad challenge, which can be seen as the goal of building machines that can see like humans, i.e., to infer or recognize general principles, situations or concepts from visual information.

In scene understanding, one relevant sub-task is focused on describe the scene in terms of labeled regions and objects. This sub-task of scene understanding can be formulated as a dense (per pixel) image labeling problem. *Dense labeling estimation* is a common problem which assigns specific values (labels) to each pixel or regions in the image. Depending on the application, the labels can represent different values, from adding extra magnitude per pixel (such as depth information) to semantic information (such as the object category).

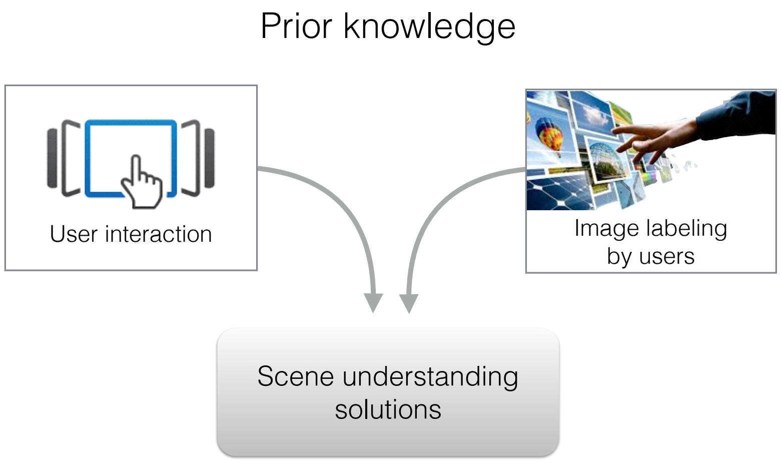


Figure 1.2: Different strategies to acquire user knowledge information used in the scene understanding tasks presented in this thesis.

Tackling scene understanding as a common and general task for any domain is unfeasible. Therefore, there are usually specific techniques and algorithms limit themselves to specific subdomains. Even in the case of limited subdomains, understanding a scene from scratch is really daunting. However, with the human perception as inspiration, we learn that humans do not start from scratch, but they rather start from a specific prior knowledge and learn more complex models from it. This thesis follows such approach: learning complex scene information from pieces of user prior knowledge. In particular, this thesis deals with two sources of prior information: interactive human input and pre-existing examples of semantically labeled data (Fig. 1.2). These two sources of information are directly related to the following two challenges, which motivate the different lines of work along this thesis.

1.2.1 Dense labeling from interactive user-labeling propagation

As previously mentioned, the prior information required for dense labeling may come from different sources. A common strategy is to design human-in-the-loop approaches [6–8], which get prior information from user interaction.

Many complex problems in computer vision and advanced image editing techniques often rely on specific per-pixel information, i.e., require labeling each pixel as a preliminary step. There are plenty of algorithms that provide good semantic segmentation estimations, for example for autonomous driving domains [9] or for medical segmentation [10]. However, in many cases, these solutions provide incomplete or noisy segmentation maps which are not accurate enough for many applications such as reilumination or augmented reality.

For the particular case of advanced image editing, techniques often require to include scene knowledge into the editing pipeline, in which such advanced per-pixel information enables more sophisticated edits.

This scene information may come from heuristic computer vision approaches but more complete results can be obtained with hybrid approaches in which both human interaction and computer vision interactively refine the required knowledge. There is previous research on interaction to propagate complex magnitudes such as color for black and white images [11], light field edits [12] or shading and reflectance in images [13]. The challenge in such cases is to keep the computational part of the process within a short time frame in order to interactively give feedback to the user interaction.

1.2.2 Dense labeling from semantic segmentation models

Other common strategy to learn from user prior knowledge is to design supervised machine-learning approaches. Standard algorithms in machine-learning require numerous training examples to yield similar results to the recognition capabilities of a person.. In the last years, techniques based in deep learning have been pushed the state of the art in many computer vision related applications [9, 14, 15]. These techniques get to solve very complex problems, that years ago seemed unsolvable, partly thanks to the availability of extremely large amounts of examples. In particular, we have witnessed significant improvements towards solving multiple tasks related to scene understanding. This thesis explores how to improve existing techniques to learn new semantic labeling models.

Semantic image labeling is crucial for autonomous systems to operate in real-world scenarios. Applications such as autonomous driving have already shown to benefit from an initial semantic segmentation of the scene, for instance, to locate drivable regions or identify obstacles and avoid collisions [16, 17]. Semantic segmentation algorithms assign one of the preset class labels to every pixel in an image. Each algorithm is often targeted to a particular task and in consequence, the set of pre-defined classes is specific and significant for such task.

Convolutional Neural Networks (CNNs) have demonstrated excellent performance on semantic image segmentation. Although successful CNN models are typically shared with the research community, not every concept or object that we need to identify for different applications is already represented in those pre-trained models. Due to a lack of resources or data, training new CNN models from scratch it is not always feasible. Thus, reusing the existing models by adapting them to the new target domain is worth to pursue and has been shown very successful in the computer vision community [18, 19].

1.3 Goals and contributions

The most relevant contributions of this thesis can be summarized in the following two groups, each of them related to one of the previously mentioned challenges considered in this work: incorporating user knowledge directly from interactions and incorporating user knowledge through learning systems which use supervised techniques built from user labeled examples.

User-labeling propagation

The first block of contributions of this thesis is related to how to integrate information from user interaction inputs into a dense labeling system. The proposed system facilitates plenty of applications that require dense labeling solutions with strict response time requirements. Plenty of complex image editing techniques require certain per-pixel property or magnitude to be known, e.g., simulating depth of field effects requires a depth map.

Taking advantage of user interaction, this thesis presents an efficient interaction paradigm that approximates any per-pixel magnitude from a few user strokes by propagating the sparse user input to each pixel of the image. The propagation scheme is based on a linear least squares system of equations which represents local and neighbouring restrictions over superpixels. After each user input, the system responds immediately, propagating the values and applying the corresponding filter. The interaction paradigm is generic, enabling image editing applications to run at interactive rates by changing just the image processing algorithm, but keeping our proposed propagation scheme.

We illustrate the benefits of the proposed system by implementing and integrating it on three interactive applications: depth of field simulation, dehazing and tone mapping. These contributions are described in Chapter 2 and Chapter 3.

Associated publications and results:

- [20] Cambra, A. B., Murillo, A. C., and Muñoz, A. (2017). A generic tool for interactive complex image editing. The Visual Computer, 1-13.
- [21] Cambra, A. B., Muñoz, A., Guerrero, J. J., and Murillo, A. C. (2016). Dense Labeling with User Interaction: an Example for Depth-Of-Field Simulation. In British Machine Vision Conference.

- [22] Cambra, A. B., Muñoz, A., Murillo, A. C., Guerrero, J. J. and Gutierrez, D. (2014). Improving Depth Estimation Using Superpixels. CEIG, 49-58
- Besides the technical publications, the code of the proposed framework is available online [2] and also there are multiple multimedia files demonstrating these contributions to facilitate their dissemination [3].

Semantic segmentation

Semantic scene understanding is an important task for robots operating autonomously in real-world applications. Recent deep convolutional neural networks (CNNs) have demonstrated to be an effective approach for semantic image segmentation, especially for tasks where plenty of labeled data is available. However, many applications need to learn new specific classes but do not have a lot of labeled training data.

The second part of this thesis addresses the problem of transferring the knowledge from existing CNN models, e.g., from autonomous driving applications, to different classes and domains, e.g., different robotic platforms. This work explores the two common transfer learning approaches for the particular problem of semantic segmentation: 1) fine-tuning existing models with the new training data, following a standard pipeline; 2) training a superpixel classifier using our proposed superpixel representation, which combines local and context information.

We evaluate the two proposed approaches on three varied binary segmentation use cases from different domains. Besides, we propose further improvements to this kind of pipelines, by adding pre-processing or post-processing modules to deal with noisy inputs or outputs. These contributions are described in Chapter 4.

Associated publications and results:

- [23] Cambra, A. B., and Murillo, A. C. (2011) Towards robust and efficient text sign reading from a mobile phone. IEEE International Conference on Computer Vision Workshops (ICCV Workshops).
- [24] Cambra, A. B., Muñoz, A. , and Murillo, A. C. How to transfer an autonomous driving model for semantic segmentation to other domains? In Robot 2017: Third Iberian Robotics Conference. (In Press)
- [25] Alonso, I., Cambra, A. B, Muñoz, A., Treibitz, T., and Murillo, A. C. (In Press). Coral-Segmentation: Training Dense Labeling Models with Sparse Ground Truth. In First Workshop on Visual Wildlife Monitoring, held with ICCV 2017.
- Besides the technical publications, the trained models and multimedia files demonstrating these contributions are available online [4].

[2] https://github.com/anacambra/app_lensblur/tree/TVCJ
[3] https://www.youtube.com/watch?v=Fps5SasG9v4
[4] http://webdiis.unizar.es/~acambra/

- Additionally, I have collaborated in the supervision of a Master Thesis project:

  Semantic segmentation with deep learning models and sparse or weak labels (student: Alonso, I.), which resulted in the previously mentioned publication [25].

1.4 Outline

The thesis is organized in two parts: the first part presents an efficient interaction paradigm to learn new scene information from user interaction; while the second part is focused on how to learn new scene information from pre-existing semantic segmentation models.

Chapter 2 proposes a new pipeline to achieve dense labeling in a very simple and efficient way, faster and with more flexibility than related approaches. Chapter 3 demonstrates how our pipeline is suitable for interactive applications developing three interactive application for simulated effects from a single image; in particular depth of field, dehazing and HDR tone mapping.

Chapter 4 proposes solutions to learn new specific classes from pre-existing semantic segmentations models. The solutions are based on transferring the knowledge from existing CNN models to different classes and domains using two common strategies: fine-tuning a previous model; using model to obtain image features and use them to group related image regions.

Chapter 2

Interactive labeling propagation

Many complex problems in computer vision require labeling each pixel in the image. When dense labeling is required in interactive settings, an efficient formulation is essential. This chapter proposes a method to achieve dense labeling in a very simple and efficient way, faster and with more flexibility than related approaches. An initial superpixel graph is used and its constraints are reformulated as a sparse linear system of equations, which is efficiently solved as a linear least squares problem. The experiments show that this method obtains comparable results for a dense depth estimation against related approaches, while providing a more generic and powerful representation of the problem. The proposed dense labeling system opens new opportunities to design interactive applications that require dense labeling estimation of any other image property.

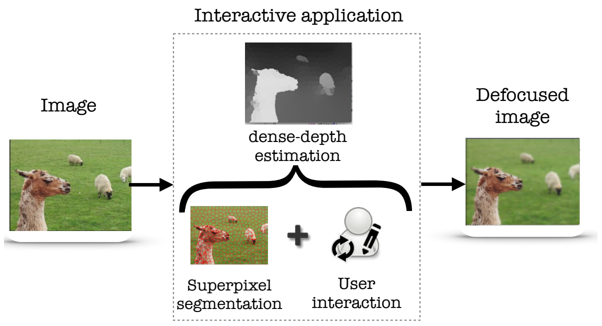


Figure 2.1: Summary of an interactive application for depth-of-field simulation on a single image. A depth map is built automatically with the proposed dense labeling approach. This depth map is used to blur the input image simulating a depth-of-field effect. The user can refine depth, focus point and aperture, and see the changes at interactive rates.

2.1 Introduction and Related work

Plenty of problems in computer vision and image processing applications rely on *labeling* techniques. These techniques consist of the following: given a set of initial values (labels) for a few pixels, an estimation process assigns the best value for every image pixel. There are plenty of applications in the literature that can be formulated in this form, and solved applying this kind of techniques, such as obtaining depth information in panoramic images [26], semantic labeling of RGB-D images [27], assigning semantic and geometric labels in conventional images [28] or interactive tools to assist the user in manual image filtering and editing [11, 29]. The work presented in this chapter is focused on this latter group of interactive applications.

One of the main challenges in this type of application comes from the fact that common dense labeling techniques present too high computational cost for interactive applications, where the feedback to the user must be at interactive rates. Therefore, specially relevant to our work are other image editing applications which, using different techniques, attempt to provide an interactive feedback to facilitate intuitive editing, e.g., to manipulate the material appearance of objects [30,31] or to separate a video into its reflectance and illumination intrinsic images [32]. Next, we briefly discuss the rest of related topics and concepts for the work presented in this chapter.

Superpixel segmentation. Many related works take advantage of the use of superpixels to help with different labeling problems, such as estimating dense depth information in multiple panoramic images [26] or assigning semantic and geometric labels in conventional images [28]. Assigning the labels according to superpixels, rather than individual pixels, allows us to reduce the complexity of the dense labeling optimization although implicitly introduces an accuracy penalty.

Dense labeling. Markov Random Fields (MRF) are a common ingredient on solutions for dense labeling problems [33] but they still present too high computational cost to achieve results at interactive rates. As the proposed approach, MRF superpixel-based approaches [26–28] or other proposed approaches to improve efficiency [34], aim to improve the execution time. Differently, our formulation is less restrictive and our experiments prove that we achieve a faster response. Another important group of dense labeling solutions is inspired by the Random Walker (RW) algorithm [35–37]. The RW algorithm can be used in an an interactive segmentation tool [38] where the user marks a few pixels with an arbitrary number of labels. This technique computes the probability of each label for each pixel as the probability of a random walk starting at the pixel that first reaches a seed with that label. All probabilities may be computed analytically by solving a system of linear equations and therefore, it can be slow for user interaction when a high number of labels is used. We find multiple works towards efficient solutions, based on offline pre-computation [39], or on the use of superpixels [40].

User interaction. Human-in-the-loop approaches have been shown very successful in a large variety of computer vision problems. We find incremental learning systems from the user input, for tasks such as fine grained categorization of conventional image content [41] or activity recognition [42], and crowdsourcing based applications, such as medical applications built from crowdsourced human knowledge [43]. In both cases, as well as in our work, each user input is a very small piece of information that is not enough on its own to solve the problem, but integrated in an automatic system facilitates an efficient and high quality solution.

2.1.1 Contributions

As previously mentioned, the main contribution presented in this chapter is a novel pipeline for interactive dense labeling, which provides a framework that can be applied in any application that involves dense labeling and user interaction. The main and distinctive properties of the presented approach are summarized next:

- We propose an efficient dense labeling estimation which is particularly well suited for the use of continuous magnitudes.
- The dense labeling is formulated as a linear system of equations over superpixels and then solved as a linear least squares problem.
- The suitability of this approach is demonstrated with several interactive editing image applications, which are detailed next, in Sec. 3.

2.2 A novel interactive dense labeling approach

This section explains the proposed interactive dense labeling system, which is used to propagate per-pixel user information. Figure 2.2 summarizes its main steps. This diagram shows how the different steps are divided into *initialization*, if they can be computed just once when the input image is loaded, and *interactive*, if they are re-calculated every time the user performs an interaction with the system.

2.2.1 Problem formulation

We model the image as a graph, where the set of nodes N are the superpixels and the edges E represent the established relationships between superpixels. Given a set of labels L, a dense-labeling problem consists in assigning a label, $l \in L$, to each superpixel in the image. Assigning a label to a superpixel is equivalent to assigning the same label to all pixels inside the superpixel. Given this representation, we define a linear system of equations that can be solved at interactive rates, leading to the desired dense labeling from a sparse input.

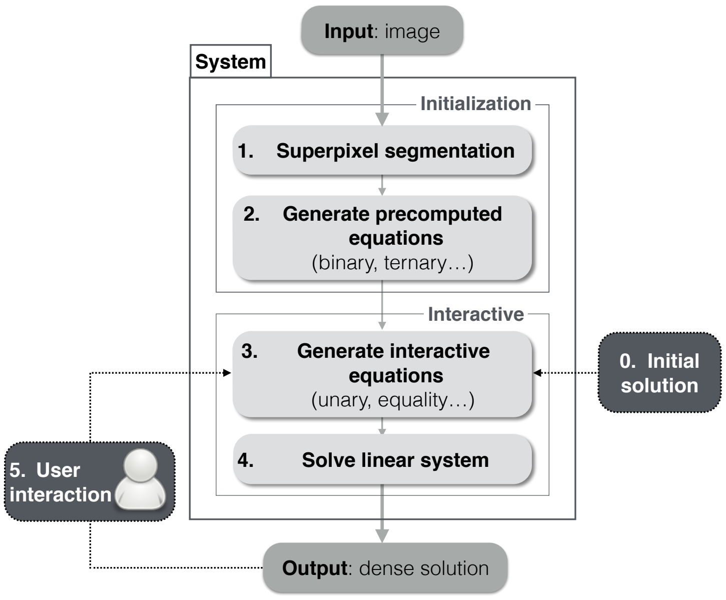

Input: image
System
Initialization
1. Superpixel segmentation
2. Generate precomputed equations (binary, ternary…)
Interactive
3. Generate interactive equations (unary, equality…)
4. Solve linear system
0. Initial solution
5. User interaction
Output: dense solution

Figure 2.2: Interactive propagation system. Steps 2 and 3 build the equations and step 4 solves the system to get the best propagation given the input image and the user interaction. Each user edition (user input) creates new interactive equations, and the output is updated, i.e., steps 3 to 5 are run, in less than 1 second.

While standard labeling formulation forces to choose a discrete set of labels, our formulation considers continuous label sets, e.g., we use real numbers in [0,1]. Our proposed solution to this labeling problem is based on a linear system of equations where the unknowns are the labels. Each equation contributes to the label value of one or more superpixels.

We generate all the equations to compose a linear system:

$$Ax = b \tag{2.1}$$

where A and b contain coefficients and independent terms respectively from all linear equations and x is the labeling solution. Since the number of equations and unknown is usually different,we find solution with a common least squares method, minimizing the error as:

$$\min_x ||Ax - b||. \tag{2.2}$$

Then, the system of Eq (2.1) is turned into:

$$(A^T A)x = A^T b. \tag{2.3}$$

The presented minimization formulation has the advantage of being linear. This allows us to split the system $Ax = b$, from Eq. ((2.1)) in the equivalent:

$$\begin{pmatrix} A_p \\ A_i \end{pmatrix} \cdot x = \begin{pmatrix} b_p \\ b_i \end{pmatrix} \tag{2.4}$$

The proposed pipeline (Fig. 2.2) enables a relatively expensive initialization step, but helps to minimize the calculations that depend on each user interaction, reaching interactive rates.

The *initialization* step first obtains superpixels and generates A_p and b_p. A_p and b_p contain the equations that are independent of the user interaction and therefore are computed only once. The included equations are just dependent on pixel positions and values, in particular, they correspond to the binary equations described in next Section2.2.2.2.

The *interactive* step generates the rest of the system, matrix A_i and vector b_i, which contains the equations related to user interactions (the unary and equality equations detailed later in Section 2.2.3.1).

2.2.2 Initialization

When the input image is loaded, we initialize the propagation system. As summarized in the previous Fig. 2.2, this involves two steps, superpixel segmentation and binary equation construction, detailed next. Even if this initialization involves some pre-computation, it is not computationally very expensive. It takes a few seconds at most (depending on processor speed and image resolution), which is within a reasonable time for the initialization of an interactive application.

2.2.2.1 Superpixel segmentation

We consider that each superpixel gets a single label value. The segmentation used is independent to our pipeline and any other implementation could be used. For superpixel segmentation we use the standard SLIC algorithm [44] with its single configuration parameter, *number of superpixels*, which depends on the image size. We estimate a superpixel size around 10×10 but the number of superpixels is limited to 1000. This algorithm is reasonably fast, although our propagation technique could use any other segmentation algorithm. This segmentation helps us to control the efficiency of the solver independently of image resolution.

2.2.2.2 Binary equation construction

In this step, we generate the matrix A_p and b_p from Eq. (2.4), which contain the equations that are independent of the user interaction and therefore are only once. The dimension of the square matrix and the vector is N, where N is the number of superpixels. By calculating and storing both the matrix and the vector during initialization, we avoid doing the matrix product (potentially time consuming) during user interaction.

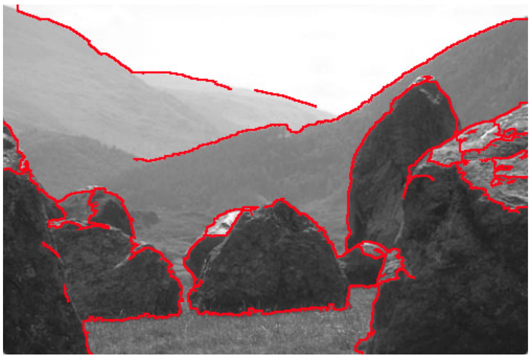


Figure 2.3: The preconditioning factor w_c in binary equations prioritizes connections whose border pixels are similar in the CIE-Lab color space. Those that exceed the D_{MAX} threshold are marked as red. It can be observed how those preserve object boundaries.

These equations establish binary relationship between values of connected superpixels. We consider that two superpixels are connected following 8-neighbors connectivity. Given two connected superpixels p and q, their binary relationship is represented as

$$w_b(x_p - x_q) = 0, \tag{2.5}$$

where x_p and x_q are the unknown values of superpixels p and q respectively.

The preconditioning factor w_b prioritizes the connections whose border is similar in the CIE-Lab color space and is defined as:

$$w_b = \begin{cases} w_c & \text{if } d_{pq} \leq D_{MAX} \\ 1 - w_c & \text{otherwise,} \end{cases} \tag{2.6}$$

where $w_c \in [0,1]$, and D_{MAX} is and fixed experimentally in 0.05 ($d_{pq} \in [0,1]$). Figure 2.3 shows that this limit respects the object boundaries. The boundaries that exceed this limit are painted in red. d_{pq} represents the color similarity between the boundary pixels of both superpixels p and q. It is defined as:

$$d_{pq} = \sqrt{(\bar{B}^L_{pq} - \bar{B}^L_{qp})^2 + (\bar{B}^a_{pq} - \bar{B}^a_{qp})^2 + (\bar{B}^b_{pq} - \bar{B}^b_{qp})^2}, \tag{2.7}$$

where $\bar{B}^L_{pq}$ is the mean of the luminance L channel of the pixels inside superpixel p which are in the 8-neighbor boundary with superpixel q. $\bar{B}^L_{qp}$ represents the opposite: pixels inside superpixel q which are in the boundary with p. The definitions for the chrominance channels a and b are analogous. All channels are normalized between 0 and 1. w_c modulates the effect of the CIE-Lab color similarity over the labeling propagation. Lower values of w_c tend to ignore color boundaries and therefore blur the whole dense labeling, while higher values (close to 1) may isolate certain superpixels. We experimentally set this value to $w_c = 0.99$.

2.2.3 Interactive propagation

Once the initialization is finished, the system starts to run the interactive part in a loop, until the user considers the interaction is finished. This interactive part is in charge of propagating of the currently available information, and consists of two steps: the addition of equations that correspond to user interaction and the value propagation itself, i.e., solving the linear system.

2.2.3.1 Equations to model user interaction data

Unary equations. The unary equations link user input with superpixel values and are built during the *interactive* steps. The user chooses a brush that corresponds to a specific value of the magnitude of interest v. For each pixel affected by the stroke inside superpixel p, we include the following equation into the system:

$$w_u x_p = w_u v, \tag{2.8}$$

where x_p is the (still) unknown magnitude value of superpixel p and $w_u = \frac{1}{\#u}$ (where $\#u$ is the number of unary equations) is a preconditioning factor that ensures stability on the behavior of the system, no matter the number or length of user strokes. We interactively add (2.8) both into the pre-computed matrix A_i and the vector b_i, as only one of the cells of the matrix and the vector is affected by the equation. Therefore, there is no need to recalculate the matrix product.

Depending on the user strokes, the user may include contradictory equations for the same superpixel. This is expected and supported by the approach, because the solver is a linear least squares minimization that will find the optimal x_p that minimizes the contradiction. Furthermore, larger strokes may include many equal unary equations that affect the same superpixel, which in practice increases the overall weight of the equation in the minimization. This is expected and desired as well.

Equality equations. While unary equations set specific superpixel values, *equality* equations establish similarity relationships between superpixels through user strokes. Those superpixels are not necessarily contiguous. Such equations are set during the *interactive* step. Given two superpixels p and q, their binary relationship is represented as

$$w_e(x_p - x_q) = 0, \tag{2.9}$$

where x_p and x_q are the unknown values of superpixels p and q respectively and $w_e = \frac{1}{\#e}$ (where $\#e$ is the number of equality equations) is the preconditioning factor. Such equations are useful where specific superpixel values are rather unintuitive (rendering unary equations useless) but similarity can intuitively be spotted.

2.2.3.2 Linear system solving

After the unary and/or equality equations are stored we solve the equation system in (2.3). The $(A^T A)$ matrix is symmetric by definition and positive semi-definite. Therefore, the system can be solved applying Cholesky decomposition, specifically LDLT decomposition, with the advantage of being fast and numerically stable. Note that all preconditioning factors (w_b, w_u and w_e) are global parameters which help to control the effect of each equation type. In order to keep such global weights constant, each preconditioning factor is related with the number of equations, so each new equation gradually reduces the influence of all the equations of the same type. This is intended and helps to preserve the stability of the global effect of each user stroke.

2.3 Quantitative analysis of the label propagation obtained

This section evaluates the proposed interactive propagation system with a quantitative and exhaustive analysis of the performance of our approach.

2.3.1 Execution time.

An essential goal of the presented labeling approach is to guarantee the response time requirements for applications that interact with a user. Table 2.1 shows the execution time of each pipeline step in a typical execution measured in a desktop computer (Intel Core i5 2,5 GHz) with a sample image.

Table 2.1: Typical execution time per step with an image of 2008x1340.

Step	**Time** (seconds)
Initialization	
1. Superpixel segmentation (950)	2.39
2. Add binary equations (2725)	0.08
Interactive propagation	
3. Add all user equations (819)	0.015
4. Solve linear system	0.20

The *initialization steps* are executed only once at the beginning, while the image is being loaded to the application. The superpixel segmentation is the most expensive step from this stage, although the segmentation used is independent from our pipeline and any other implementation could be used. The number of superpixels defines the number of unknowns in the linear system and as consequence, it determines the propagation speed. We can adjust the number of superpixels depending on the image resolution in order to keep the interactive execution time.

The *interactive loop steps* (building new unary equations according to new user input, solving the system and re-estimating the depth map) are executed after each user interaction. For example, in a typical image of 2000×1340 (with 950 superpixels), our system can propagate the user information in 0.20 seconds. This time includes the time for generating the user equations, solving the system and building the dense map solution.

2.3.2 Interactive dense labeling evaluation

This section presents a numerical validation of our interactive dense labeling system. We focus the validation measurements on the part of the pipeline that estimates the labeling (step 4 from Fig. 2.2), since it is the only common part in all the approaches compared.

We compare our results with state-of-art dense labeling approaches in two different settings, each of them using very different types of input, as detailed in Fig. 2.4. The first one is a dense but unreliable input labeling obtained automatically from two stereo images and the second one is a sparse but more reliable input labeling obtained from a few user strokes.

We should note that our work is focused and targeted on the second setting, and it aims to generate a good estimation using just a few pieces of information from the user (input detailed in Fig. 2.4(b)). However, MRF-based methods are focused on obtaining an accurate solution using an available initial estimation, typically distributed all over the image (as detailed in Fig. 2.4(a)). Therefore, they are suboptimal when a very sparse input is used, as we confirmed with the results in this section. For a more complete evaluation, next experiments present results for both cases for the considered approaches.

2.3.2.1 Reference labeled data used

For these labeling experiments we use well known public data [45] [46] designed to evaluate stereo algorithms. In this dataset the ground truth labels represent the disparity between corresponding pixels from two images. We chose this dataset because it had public results for recent dense labeling related methods.

2.3.2.2 State-of-art approaches considered

We compare our algorithm to the following state-of-the-art algorithms described in a well known comparative of MRF-based dense labeling approaches [33]: iterated conditional modes (*ICM*) [47]; Graph Cut (*GC*) with α-expansion moves (*Expansion*) and $\alpha\beta$-swap moves (*swap*) [48]; two implementations (*BP-S* and *BP-M*) of loopy belief propagation [49], and Sequential three-re-weighted message passing (*TRW-S*) [50]. Besides, since our focus is on speed and interactive properties, we include the recently presented block coordinate descent algorithm (*BCD*) [34], which was developed with an emphasis on speed, and a Random Walk based approach, the implementation provided by [38], which was designed as an interactive segmentation tool. This formulation is very close to ours and it is also based on a linear equation system. Note that both MRF and RW implementations are based on pixel-wise formulations, while our work is superpixel-based, therefore it has a disadvantage over pixel-wise techniques in terms of accuracy but presents higher efficiency without much penalization in accuracy, as we can see in the following experiments.

We also include a superpixel based version of *GC*, which is the only related method whose available implementation can be directly adjusted to support a superpixel formulation. We adapt its MRF-graph edges to use superpixels as nodes, as our approach does, and adapt its MRF unary and binary cost functions to include the same constraints as our unary and binary equations.

2.3.2.3 Error measure.

To compare the quality of the different algorithms solutions, we measure the error obtained in each solution as the *mean of the differences (or mean error)* between each pixel in the solution and the same pixel in the ground truth, as follows:

$$\bar{\mu}_{\{G-I\}} = \frac{\sum_p^N |l_p^G - l_p^I|}{\sum_p^N n_p} \tag{2.10}$$

where l_p^G denotes the labeling in the ground truth and l_p^I the obtained labeling proposed.

2.3.2.4 Results

We run the experiments with two very different inputs (Fig. 2.4): a **dense** but unreliable (noisy) input labeling obtained automatically from two stereo images, and a **sparse** (but reliable) input labeling obtained from a few user strokes

Dense labeling input. As we mentioned, the MRF based methods use all the initialization information they can extract from two stereo images, by running an automatic disparity estimation algorithm for stereo. In order to evaluate our pipeline and the RW approach in similar settings than the other MRF studied approaches we need an initial disparity estimation. As initial disparity map, we use the same initial map utilized in the ICM algorithm [33], Fig. 2.4 (a). Note that this noisy and unreliable input is not the target case for our method.

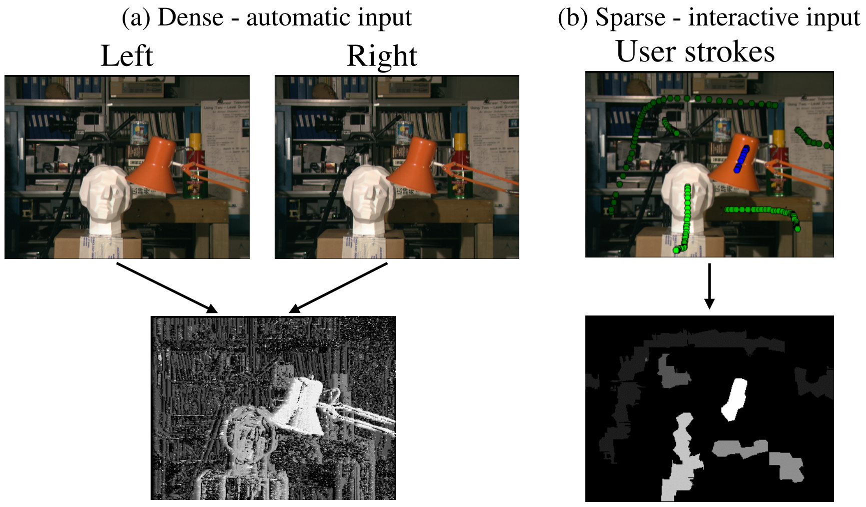


Figure 2.4: Types of labeling initialization used in our experiments.

Sparse user input. In this test, the input for all methods evaluated is the same set of pixels initialized from user strokes, Fig. 2.4 (b). As the user associates each pixel stroke with a depth value, we can create a sparse initial depth map to initialize all the methods compared here. In this experiment, the user input consists on 5 different levels of depth assigned to different user strokes. Note that the input image in Fig. 2.4 (b) highlights the whole superpixels affected by user strokes, but actually we are only including one unary equation per pixel affected.

Table 2.2: Execution time (*seconds*) and mean error (*err*) for dense labeling obtained for 3 test images (name and resolution in each column) with automatic input *(a)* and user input *(b)* initialization. *#Labels*: number of disparity levels considered on each test.

(a) AUTOMATIC DENSE INPUT

Method	**Tsukuba** (384x288) #Labels=16		**Venus** (434x383) #Labels=20		**Teddy** (450x375) #Labels=60	
	time	**err**	**time**	**err**	**time**	**err**
Pixel-based						
ICM [47]	0.520	0.12	0.460	0.10	1.900	0.13
Expansion [48]	2.220	0.02	6.940	0.02	19.90	0.05
Swap [48]	2.250	0.02	7.010	0.02	12.60	0.05
TRW-S [50]	8.840	0.02	115.0	0.02	158.0	0.05
BP-S [49]	1.370	0.02	8.690	0.03	21.20	0.05
BP-M [49]	13.30	0.02	–	–	193.0	0.05
BCD [34]	0.920	0.09	1.500	0.17	2.760	0.08
RW [38]	0.200*	0.12	0.400*	0.20	0.600*	0.16
Superpixel-based						
Expansion	3.090	0.06	6.320	0.10	6.210	0.08
Ours	**0.002**	0.06	**0.005**	0.10	**0.005**	0.09

–: the method did not converge to a solution
*: execution time is measured in Matlab.

(b) SPARSE USER INPUT

Method	**Tsukuba** (384x288) #Labels=5		**Venus** (434x383) #Labels=5		**Teddy** (450x375) #Labels=5	
	time	**err**	**time**	**err**	**time**	**err**
Pixel-based						
ICM [47]	N/A	N/A	N/A	N/A	N/A	N/A
Expansion [48]	N/A	N/A	N/A	N/A	N/A	N/A
Swap [48]	N/A	N/A	N/A	N/A	N/A	N/A
TRW-S [50]	N/A	N/A	N/A	N/A	N/A	N/A
BP-S [49]	N/A	N/A	N/A	N/A	N/A	N/A
BP-M [49]	24.40	0.14	34.20	0.09	35.10	0.18
BCD [34]	–	–	–	–	–	–
RW [38]	0.500*	0.13	0.600*	0.20	0.700*	0.09
Superpixel-based						
Expansion	4.170	0.06	6.370	0.14	7.960	0.09
Ours	**0.002**	0.06	**0.005**	0.15	**0.005**	0.06

–: can't provide to available implementation.
*: execution time is measured in Matlab.

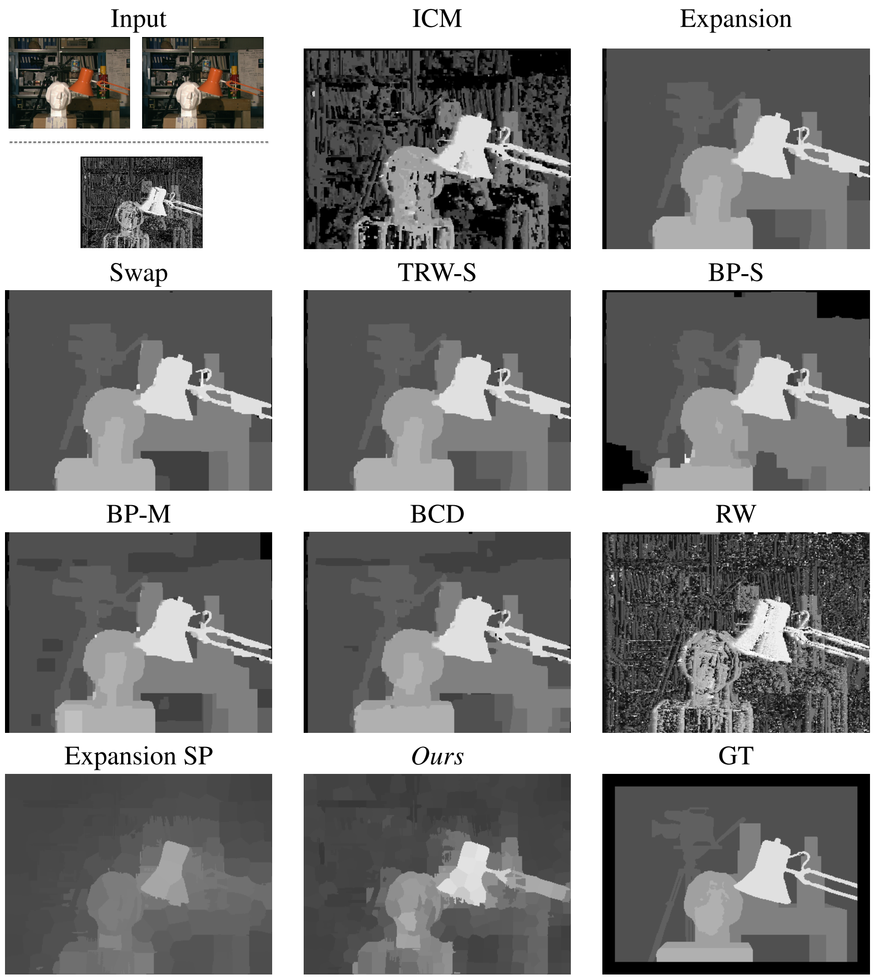


Figure 2.5: *Tsukuba*-test. Final disparity estimation obtained by each of the evaluated methods using a common *dense automatic input*. Our method is not as accurate as the best state-of-art MRF solver, better suited for dense inputs, but obtains significantly faster solution which is essential for the targeted interactive applications.

Discussion of the results. Table 2.2 shows error and execution time (measured on a standard desktop machine Intel Core i5 2,5 GHz) for the dense labeling estimation in three of the tests run. Figures 2.5 and 2.6 show the final disparity estimation obtained by each of the evaluated methods for the *Tsukuba*-test.

As previously mentioned, MRF-based methods are focused on obtaining an accurate

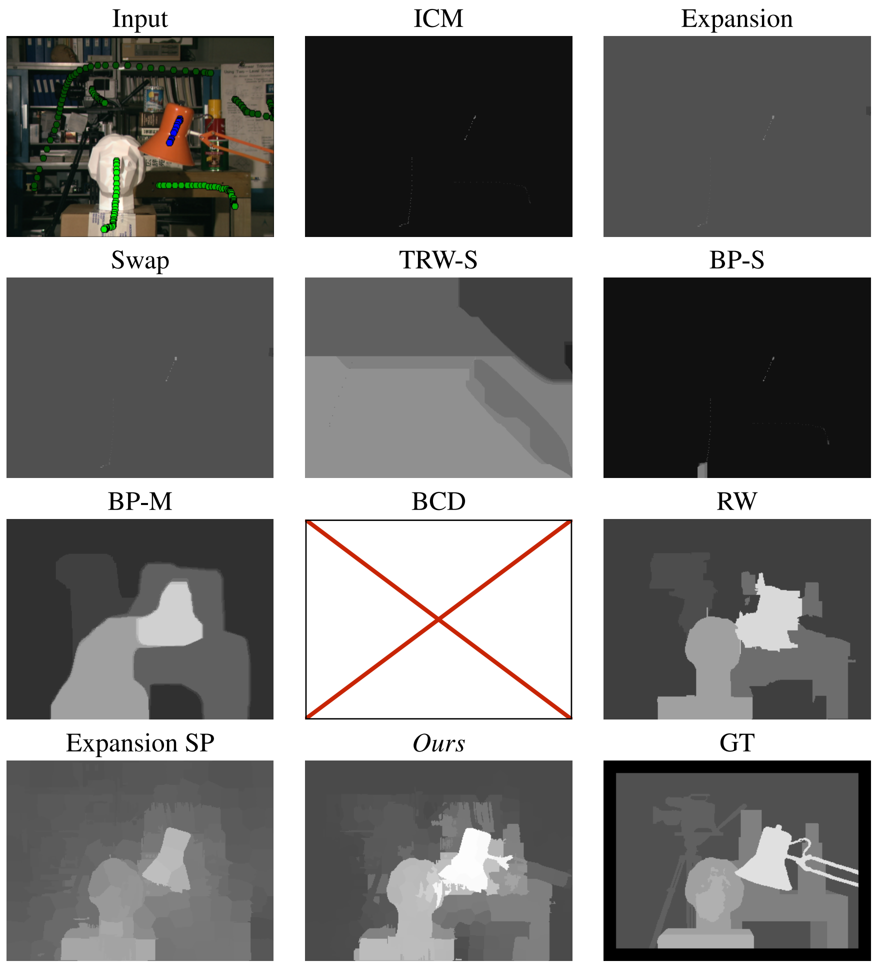


Figure 2.6: *Tsukuba*-test. Final disparity estimation obtained by each of the evaluated methods using a common *sparse user input*. Many of the studied approaches are not designed to handle such a sparse initialization, typical in the targeted interactive applications.

solution using an dense initial estimation and, as we can see in Table 2.2 (a), these methods obtain the best estimation in terms of accuracy. However, our algorithm is faster than any of the other algorithms in this comparative, while keeping comparable accuracy than the fastest ones (which still are one order of magnitude slower). Note that the direct comparison with RW implementation is somehow unfair as the available code is implemented in Matlab, while the rest use C++, so we could expect the speed up of around one order of magnitude typically observed between these two environments. Theoretically, the

RW [38] requires solving a linear system per label in the solution, while our approach only solves one, independently of the number of labels. For example, in tests of Table 2.2 (b) where we use five labels, the RW solver would be around five times slower than our approach.

Table 2.2 (b) shows our approach obtains also the best estimation in terms of accuracy, although we could say that the three best approaches obtained results of comparable quality. We could say that by definition, superpixel based approaches are typically less accurate than pixel-based methods but faster to compute. Our work is superpixel based and therefore has advantage in terms of time cost over pixel-wise techniques. However, this time advantage is achieved at a small sacrifice on accuracy, as it can be seen from the quality of final estimations obtained in the Table 2.2.

An important advantage of our method is the flexibility. First, as we use a continuous label set, we do not need to choose the number of possible final labels a priori. Second, only MRF-based methods can include in the final solution intermediate label values like us (but at much higher cost), but RW can only assign a choice among the input labels, i.e., if five depth values are given as input, only those five values will compose the final solution. Another interesting advantage of our method is that the execution time does not fluctuate a lot with image dimensions or number of labels, since is not directly depending on any of those magnitudes, but on the number of superpixels.

2.4 Analysis of parameters influence on the approach performance.

This section analyzes the influence of the main parameters of our approach on its performance. As described in previous Section 2.2, our formulation includes some control parameters which have been set experimentally to provide the best trade-off for our application. The following results analyze the influence of these parameters value on the labeling errors and present the justification for the recommended values. Note that these recommended values correspond to our particular case of dense depth estimation. If the pipeline is used for estimating a different real magnitude, they would probably change.

Figures 2.7 and 2.8 show how the estimated depth map and the error varies, respectively, as these parameters change. We measure the error obtained in each solution with regard to its ground truth. We utilized the same images used in the exhaustive evaluation of our pipeline described in following Sec. 2.3.2. Each sub-plot analyzes the effect of one isolated parameter with a fixed value for the rest. The parameters considered are: w_u, which represents the weight of unary over binary equations, D_{MAX}, the threshold for color similarity between neighbouring superpixel boundaries, and w_c, weight of binary equations that pass this color threshold over the rest of binary equations. The most suitable values are w_u=0.4, D_{MAX}=0.04 and w_c=0.99.

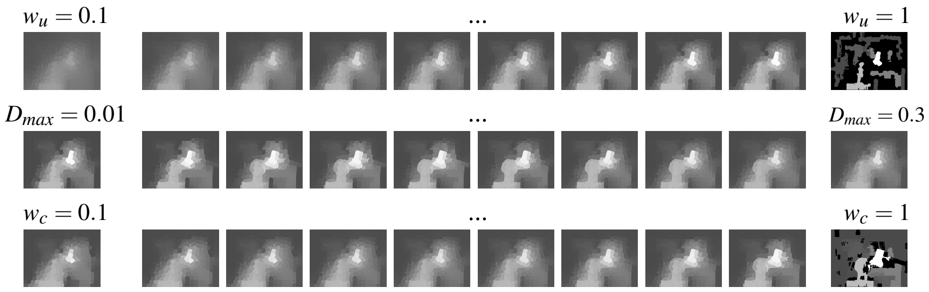


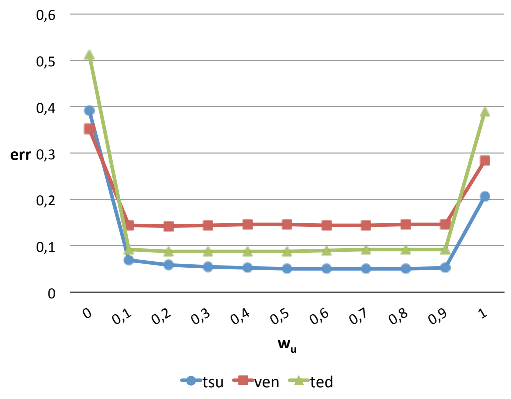
Figure 2.7: Labeling obtained with different values of the system parameters: w_u, D_{MAX} and w_c.

Unary equation weight. Figure 2.8 (a) shows the variation in the error obtained depending on the unary weight (w_u). This weight somehow determines how reliable the user input is, since we give more or less importance to the unary constraints, directly obtained from the user input. We can observe that the error is pretty similar in intermediate values, what implies that the system is not very sensitive to this value, as long as it does not become 0 or 1, i.e., using only binary or only unary constraints. Here we set the value of w_c to 0.5, since this makes all binary equations equally important regardless of their color similarity. D_{MAX} is not relevant in this case since we treat equally all binary equations.

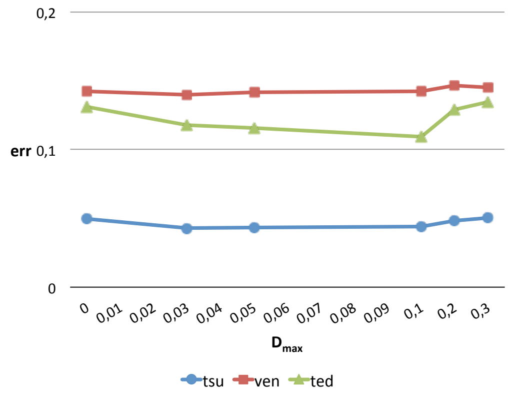
Color similarity threshold. Figure 2.8 (b) shows the error variations depending on the D_{MAX} parameter. In this test, we set $w_c = 0.9$ to highlight the influence of the color binary equations, since the D_{MAX} parameter establishes which binary equations are considered color binary equations, i.e., have a low color distance. If D_{MAX} is too high (above 0.1), the effect of w_c is dramatically reduced since almost all binary equations pass the threshold. We set the default value way below 0.1 to make sure propagation across boundaries is only encouraged if boundary color is very similar.

Binary equation weight. Figure 2.8 (c) shows the effect of w_c, which weights the importance of color binary equations over the rest. If we only connect superpixels with binary equations when their boundary color is very similar (w_c=1), the error increases because too many neighboring relationships are completely ignored. The best behaviour is observed with very high values. The default value is set to 0.99.

(a) Error varying the unary equations weight (w_u)



(b) Error varying boundaries difference threshold (D_{max})



(c) Error varying the color binary equations weight (w_c)

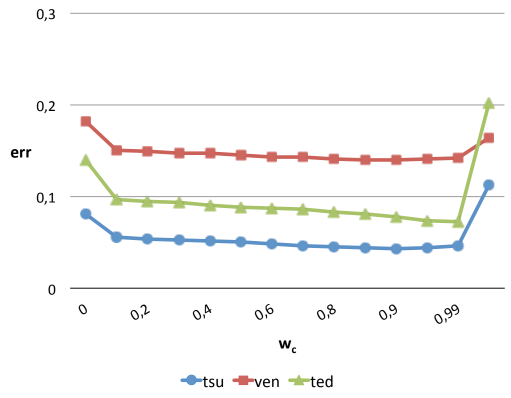


Figure 2.8: Error obtained with different values of the system parameters: w_u (a), D_{MAX} (b) and w_c (c).

2.5 Conclusions

In this chapter, we have developed an efficient approach for dense labeling based on a superpixel linear least squares formulation. Our formulation has the advantage of providing an interactive solver, which is key to interactive post-processing applications, such as the ones we present in the next chapter.

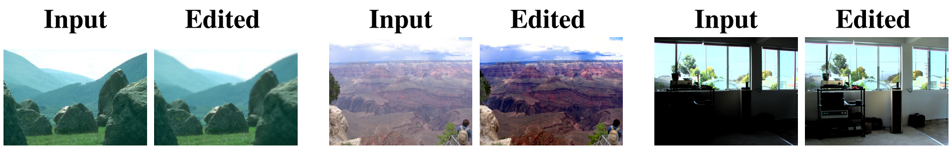
Our accuracy is limited by the number of superpixels and by the accuracy of the input, which is probably rather approximate because it is coming from the user interaction. Still, we have shown that we yield results which are accurate enough for many applications and often comparable to other slower state of the art methods. Besides, since we target an interactive technique the user can always refine and improve the input iteratively.

We believe that our approach will inspire future research for interactive editing applications based on dense labeling. The interactivity rates of our approach, together with the aforementioned expanded flexibility, provide great potential for other applications dealing with tasks such as semantic labeling, image restoration, intrinsic imaging or inpainting.

Chapter 3

Applications of interactive labeling propagation

Plenty of complex image editing techniques require certain per-pixel property or magnitude to be known, e.g., simulating depth of field effects requires a depth map. This chapter shows that our propagation system (presented in previous chapter) is generic, enabling image editing applications to run at interactive rates by changing just the image processing algorithm. We illustrate this through three interactive applications: depth of field simulation, dehazing and tone mapping.



(a) Depth of field (b) Dehazing (c) Tone mapping

Figure 3.1: Image editing examples using the proposed generic interaction paradigm. From a few user strokes our approach estimates a per-pixel magnitude that enables the advanced editing of (a) depth for a *depth of field* effect (focus point is at the middle scene elements); (b) transmittance for a *dehazing* effect; (c) luminance and brightness for *tone mapping*.

3.1 Introduction and Related work

Advanced image editing techniques often rely on specific per-pixel information about certain magnitudes, in addition to each pixel intrinsic RGB coordinates. This extra per-pixel information may come from additional knowledge or control over the capture process. For instance, depth can be estimated from sensor motion, either intended [51] or accidental [52]. Alternatively, advanced heuristics and priors may lead to a plausible (but not necessarily accurate) additional information that can be used for filtering purposes. Examples of these include heuristics to obtain depth from a single image [53], or transmittance priors for dehazing [54]. These heuristics and priors typically involve assumptions that work very well in situations that meet them. However, we typically find specific cases when they are broken, which leads to poor performance on the editing results.

Taking advantage of user interaction is another option for gathering additional per-pixel information, which does not impose any restriction regarding control of the image capture or advanced heuristics that could not be generic enough. On one hand, traditional generic user interaction paradigms (such as geometric primitives, *lassos* or color-aware *magic wands*) are not practical for complex information editing. On the other hand, other specialized editing tools define interaction paradigms that are very practical but tailored to specific applications, such as light field editing [55], intrinsic image decomposition [13] or intrinsic video decomposition [32]. Our paradigm is modeled as a propagation problem, in which the user sets some seed values and/or restrictions, with one or more strokes, which are propagated throughout all the pixels. This propagation results on a per-pixel estimation of the target continuous magnitude (continuous in value, not in image space). The propagation is formally represented as a least-squares linear system of equations over a set of superpixels. This provides us a great control over the accuracy vs. propagation time trade-off.

Image editing techniques. In this chapter we illustrate how our interaction paradigm (Chapter 2) works through three real applications: depth of field effects [52, 56], dehazing [54, 57] and tone mapping [58, 59]. Related to our work, there is previous research on interaction to propagate complex magnitudes: color for black and white images or video [11, 29, 60], light field edits [12], image segmentation [61], shading and reflectance in images [13] or video [32] and depth [62–64]. However, the interaction at each of these techniques is tailored to the specific application itself, and in some cases it does not reach interactive rates. In contrast, our work presents a global propagation technique that is oblivious to the underlying filter, is more flexible and works at interactive speeds.

Interactive editing. Examples of interactive image editing applications include transferring edits between different views through affine transform estimation [65], multiview depth transfer through shortest path algorithm [66], appearance transfer using simplex optimization for reducing parameter space [67], or material editing on a linear space using a GPU [68].

Our propagation paradigm is closely related to other image editing techniques which propagate continuous magnitudes. Lischinski et al. approach [69] solves systems at different image resolutions to show a progressive result (which otherwise would not be interactive). Chen et al. [60] propose a linear system over feature space. SteroBrush [70] also uses a linear system for propagating stereo disparities while preserving pleasantness, which becomes interactive by taking advantage of the GPU. Such linear optimizations are per-pixel, while our approach works at superpixel level, therefore becoming more efficient. The AppProp approach [71] propagates exposures through a per-pixel linear system, but it is done faster by identifying a subset of representative columns in the matrix. Instead of column sampling, previous work [72] has reduced the parameter space by clustering it into a KD tree. Still, in both cases matrix operations (inversions, products) are needed before solving the system. Our approach is inspired by such approaches, reducing the linear system by working in superpixel space, but improves over them by building the linear system matrix on the fly.

Iizuka et al. [63] propose propagation through optimization based on geodesic distances over superpixels (with some additional filtering), with a propagation speed that matches our work. However, their edits are limited to such distances, while our formulation is more flexible and enables more global edits.

3.1.1 Contributions

We work towards a more general approach for advanced user interaction paradigms. Whenever certain filter needs additional per-pixel information, it is densely generated by our approach from very sparse and intuitive user interaction. Previous chapter has illustrated this user interaction by means of simple user strokes for the specific case of depth estimation. Built on top of that, we elaborate it into a very simple and intuitive interaction paradigm, inspired and distilled from previous specific image editing applications, which is generic and oblivious to the underlying image operator. Other contributions are several interactive applications, extensions of several state-of-the-art image filters, that now benefit from our interaction scheme.

3.2 Applications

In this chapter, we illustrate the versatility of our interactive propagation (Chapter 2) scheme through several applications described next. Figures 3.2, 3.3 and 3.4 show the behavior of each application: depth of field, dehazing and HDR tone mapping, respectively. The available on-line video [1] demonstrates the real time execution and the behavior of each application. Appendix A includes more additional examples obtained with our three interactive applications.

[1] https://www.youtube.com/watch?v=Fps5SasG9v4

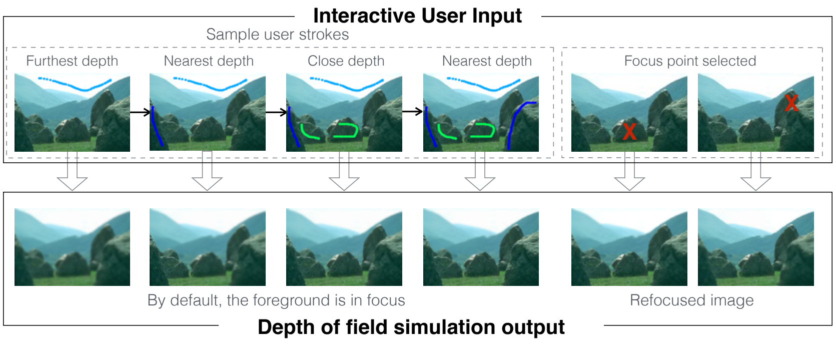


Figure 3.2: Left: the user marks a few strokes (different colors for different depths) to represent object positions. Each new edition interactively propagates depth estimation and applies a depth of field effect. Right: the user changes the focus point.

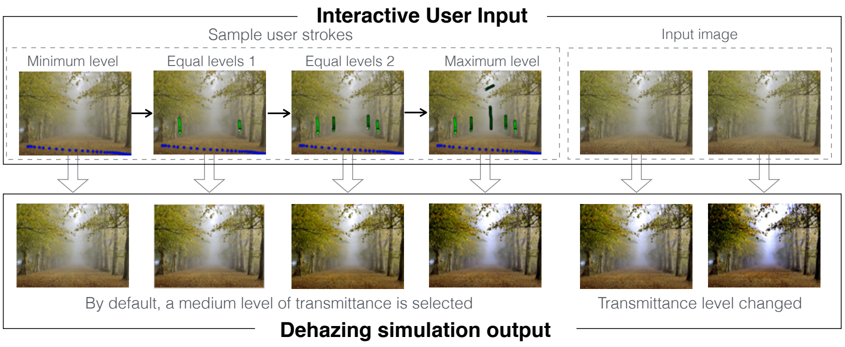

Interactive User Input
Sample user strokes
Input image
Minimum level
Equal levels 1
Equal levels 2
Maximum level
By default, a medium level of transmittance is selected
Transmittance level changed
Dehazing simulation output

Figure 3.3: Left: the user marks a few strokes (different colors for different transmittance levels) to represent the level of fog. Each new edition interactively re-estimates transmittance values and applies the dehazing effect. Right: the user can adjust the level of the dehazing effect.

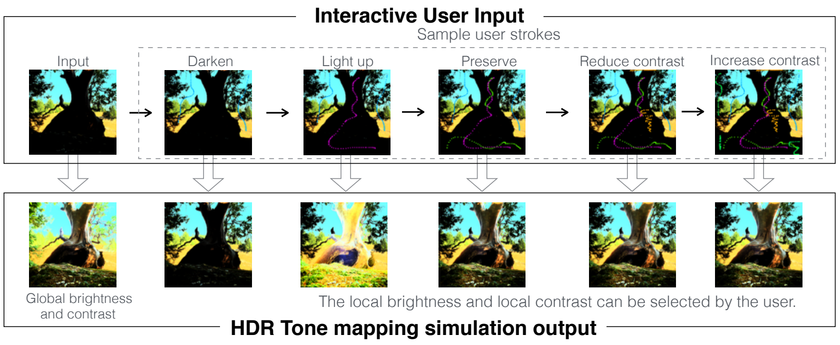
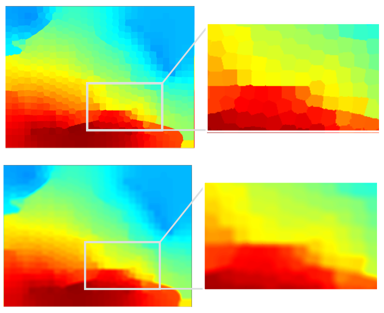

Interactive User Input
Sample user strokes
Input
Darken
Light up
Preserve
Reduce contrast
Increase contrast
Global brightness and contrast
The local brightness and local contrast can be selected by the user.
HDR Tone mapping simulation output

Figure 3.4: Left: the user can adjust global parameters. Right: the user marks with strokes that regions need to be darken or illuminated and how the contrast of the regions should increase or reduce. Each color represents a different action.

Interactive propagation. As we mentioned, our proposed system is built upon prior work, presented in previous chapter, and which is summarized in Figure 2.2. In the interactive propagation, we have included an extra step, which is the application of the image filter that uses the corresponding propagated magnitude. Due to superpixel segmentation, the final propagated values could be locally coarse and cause artifacts in processed image when the corresponding image filter is applied. To avoid this possible artifacts (similarly to related work [63]), we apply a **bilateral filter** to the propagated values. Fig. 3.5 shows the effect of applying such bilateral filter in our dehazing application.


(a)

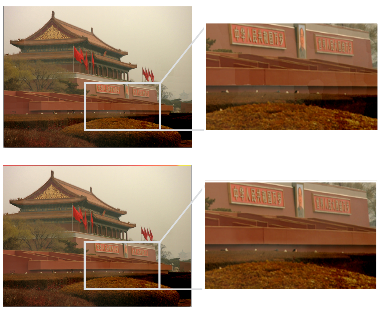
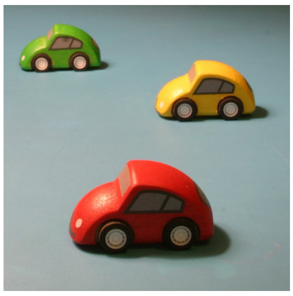
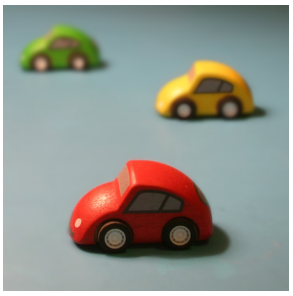

(b)

Figure 3.5: Effect of applying a bilateral filter to the transmittance map (a) used in our interactive dehazing application (b). Bilateral filter avoids artifacts caused by the superpixel segmentation in the processed image.

3.3 Depth of field


(a)


(b)

(c)

Figure 3.6: Interactive depth of field application. (a) Input image. (b) Processed image after applying the depth of field effect. (c) Depth map used to apply the effect (estimated by our application from user interactions).

In photography, control over in-focus and out-of-focus elements is an important artistic and expressive tool. **Depth of field** effect consists of the blurring of out-of-focus objects on a single image from a depth map as software post-processing. This depth can be estimated from camera motion, either accidental [52] or fixed to capture two images (stereo) and then estimate depth from them [51]. Figure 3.6 shows an example of the interactive depth of field application.

In our work, depth is obtained from the user by means of our propagation approach, as simple user strokes (using four different brushes associated with different depths in the range $[0, 1]$). Then, we obtain the depth of field effect by applying a set of convolutions (blurs) of different radius according to the difference between depth at each pixel and focal distance, following previous work [56]. The blurs are applied up to a maximum radius defined by the user. The focal distance is set by the user.

Figure 3.6 shows an example of this application, where the focal distance is fixed at the front scene elements. Therefore, elements from the back of the scene (according to the estimated depth map) become blurred after the processing. Figure 3.2 includes several intermediate outputs while processing another example image. The output is automatically refined as additional sample depth strokes are provided by the user. The user can also change the desired focal distance by clicking on the desired focus point.

3.4 Dehazing

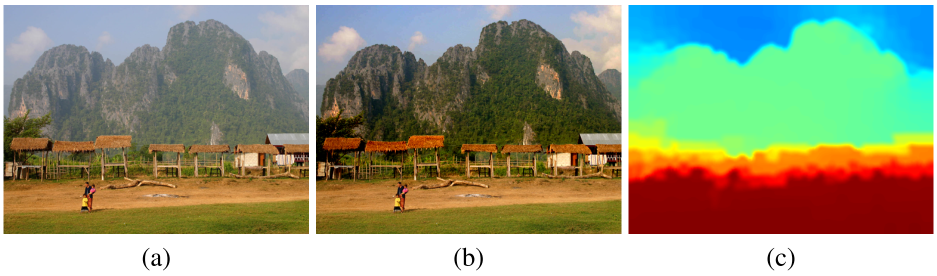


Figure 3.7: Interactive dehazing application. (a) Input image. (b) Processed image after applying the dehazing effect. (c) Transmittance map used to apply the effect (estimated by our application from user interactions).

As light interacts with small particles suspended in air (or water), fog, smoke or the aerosols in the atmosphere become visible. Repeated interactions across such media reduce the visibility by creating a translucent layer of ambient light. **Dehazing** consists on removing such ambient light from a single image. For that purpose, algorithms estimate the medium's transmittance (visibility) through heuristics and priors such as interpreting albedo as locally constant (color lines) and transmittance as smooth [54] or globally identify the haze-free colors along haze lines [57]. This previous work models haze as:

$$I(\mathbf{x}) = t(\mathbf{x})J(\mathbf{x}) + (1 - t(\mathbf{x}))A, \tag{3.1}$$

where $\mathbf{x}$ are the 2D pixel coordinates, $I(\mathbf{x})$ is the input hazy image, $J(\mathbf{x})$ is the dehazed image (what we want to obtain), $t(\mathbf{x})$ is a scalar *transmittance map* (unknown, $0 \le t(\mathbf{x}) \le 1$) and A is the ambient light (a RGB vector, unknown).

In our case, the transmittance map $t(\mathbf{x})$ is propagated from user strokes using our technique, and A is approximated as the average of all pixels $\mathbf{x}$ where $t(\mathbf{x}) > 0.9$ (the most occluded pixels). Then the image is easily obtained from (3.1) as

$$J(\mathbf{x}) = \frac{I(\mathbf{x} + (t(\mathbf{x}) - 1)A}{t(\mathbf{x})}. \tag{3.2}$$

Figure 3.7 shows an example of this application, where the fog is removed from the scene according to the estimated transmittance map. Figure 3.3 illustrates incremental output results, which are refined interactively as the user provides additional strokes to mark the level of fog in different parts of the image.

3.5 Tone mapping

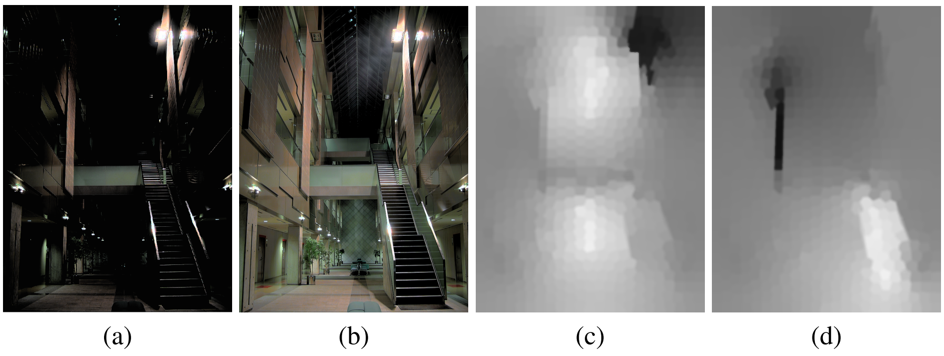


Figure 3.8: Interactive HDR tone mapping application. (a) Input HDR image. (b) Output LDR image after applying the tone mapping using the estimated local brightness map (c) and local contrast map (d).

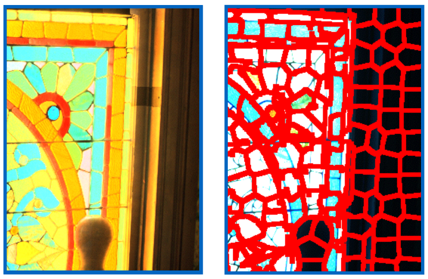
The potential range of luminance in the real world is rather large, while devices can only capture or reproduce two orders of magnitude within that range. However, High Dynamic Range image formats can represent (with floating point representation) all potential luminance values, and can be composed from several images or generated synthetically. **Tone mapping** is the process of converting a High Dynamic Range (HDR) image into Low

Dynamic Range (LDR), representable by any device. Their effect can be either global (equal to every pixel) or local (often adapting global algorithms with convolution kernels) [58]. For our tone mapper we follow the following tone curve [59] that compresses the luminance of the HDR image L_{HDR} into the luminance of the LDR image L_{LDR}:

$$\begin{aligned} L' &= \log(L_{HDR}) - b \\ L_{LDR} &= \begin{cases} 0 & \text{if } L' \leq -d_l \\ \frac{c}{2}\frac{L'}{1-\left(c-\frac{1}{d_l}\right)L'} + \frac{1}{2} & \text{if } -d_l < L' \leq 0 \\ \frac{c}{2}\frac{L'}{1+\left(c-\frac{1}{d_h}\right)L'} + \frac{1}{2} & \text{if } 0 < L' \leq d_h \\ 1 & \text{if } L' > d_h \end{cases} \end{aligned} \tag{3.3}$$

where d_l and d_h are the lower and higher midtone ranges (set up to the recommended values of 2 and 1, respectively) and b (brightness) and c (contrast) are the tone mapper parameters of the tone curve. In our case, for the sake of simplicity, we do not apply color correction. Also, we apply this curve per-pixel, where brightness and contrast are obtained from propagation: the user can set, in the image local values for brightness and contrast, that are propagated through the whole image in two magnitudes (as opposed to previous applications, that propagated just one).

Figure 3.8 shows an example of this application, where the input HDR image is converted to a LDR output image. Figure 3.4 illustrates incremental output results. The user can adjust the global image levels of brightness and contrast, but the user strokes define local values of brightness and contrast for different regions in the image.


(a)

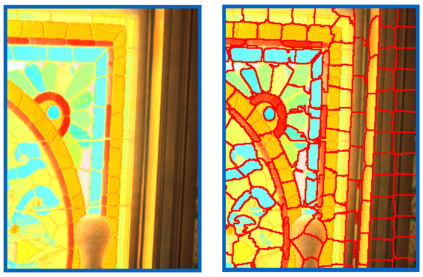

(b)

Figure 3.9: Superpixel segmentation of (a) HDR image and (b) LDR image.

Note that the algorithm we use for superpixel segmentation is not particularly well suited to work with HDR images. It does generate a superpixel segmentation, but as shown in the example from Fig. 3.9(a), the superpixel boundaries do not correspond well with scene objects. This limitation is accentuated in regions with very few or contradictory user input values. Then, the propagated magnitude suffers strong transitions between close superpixels creating artificial visual boundary artifacts. This could sometimes be solved with additional user input. However, a simpler solution is to apply this filter in two steps. First we get an initial (noisy) superpixel segmentation on the HDR image, used to apply an initial tone-mapping filter that generates an intermediate LDR image. Then, we propose to re-compute the superpixel segmentation over this intermediate LDR image, where we obtain a better superpixel segmentation that can be used to obtain a better final result without artifacts (see Fig. 3.9(b)).

3.6 Results

In previous Section 2.3, we have formally validated that our scheme can propagate interactively a few sparse sample values of a continuous magnitude to all image pixels. It is efficient while obtaining comparable quality with respect to related approaches. This section presents a set of representative results obtained using the three interactive applications presented before, in order to analyze different situations, limitations and advantages on real applications.

3.6.1 Execution time

Efficiency is a key contribution of the proposed interaction paradigm, as it guarantees response time requirements for applications that interact with a user. Note that, with each user stroke, the propagation is calculated and the filter is applied. The cost of our three filter algorithms is linear with respect to the number of pixels. In typical images of 2000×1340 (with 950 superpixels), the interactive steps can be applied in around a second. The three implemented applications are available online [2] and work at interactive rates in a regular desktop computer (Intel Core i5 2,5 GHz).

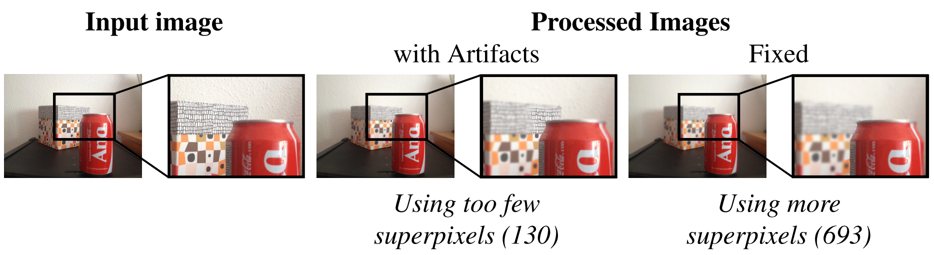


Figure 3.10: Example where superpixel segmentation issues produces artifacts in the processed images of our interactive depth of field application. Artifacts are caused when the superpixel boundaries do not correspond with the object boundaries. When superpixel segmentation is improved, these artifacts are not visible in the processed image.

[2] https://github.com/anacambra/app_lensblur/tree/filters

3.6.2 Limitations

Superpixel based approaches are faster to compute but typically less accurate than pixel based methods. Efficiency is achieved at a small sacrifice on accuracy of the final propagation and therefore, it affects the result obtained when the filter is applied to specific images. Figure 3.10 shows an example of the depth of field application. The focused object is the *red can*. As we can see, in the processed image with artifacts, part of the *box* is incorrectly focused (the *box* is behind the *red can* and therefore should be defocused). This artifact is produced by insufficient or incorrect superpixel segmentation, i.e., there is a misalignment between superpixel and actual object boundaries. These artifacts can be easily eliminated increasing the number of superpixels used.

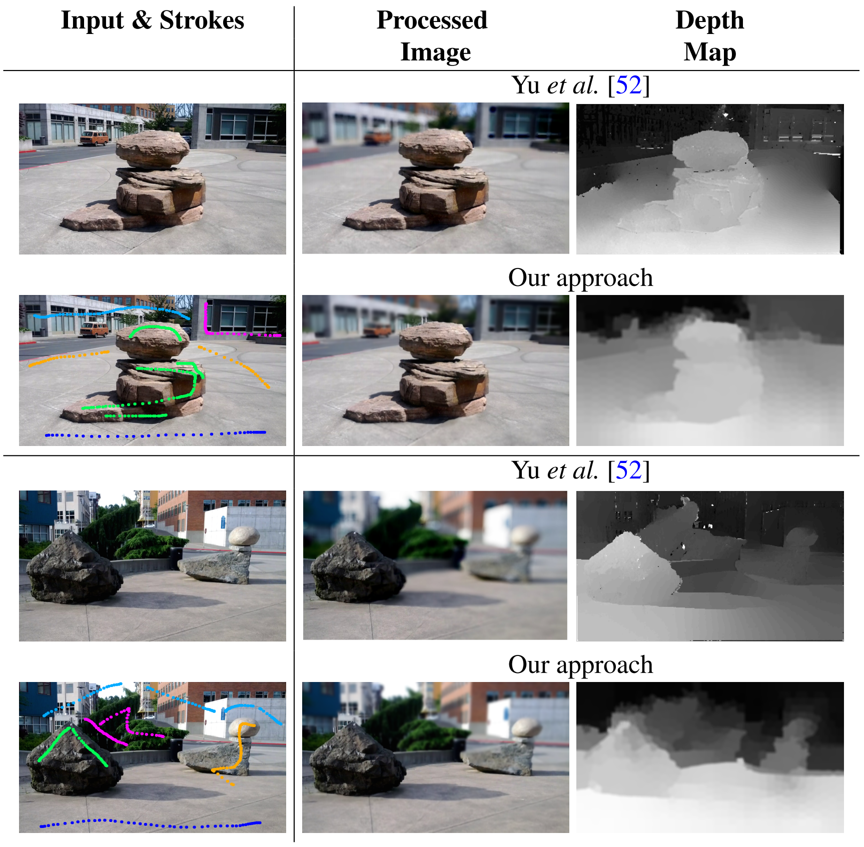


Figure 3.11: Comparison of our results and a well known algorithm for depth of field effect. Depth map estimated and final processed image are similar, but our method works with a single image and very simple input (strokes), as opposed to Yu *et al.* work, which requires a sequence of images.

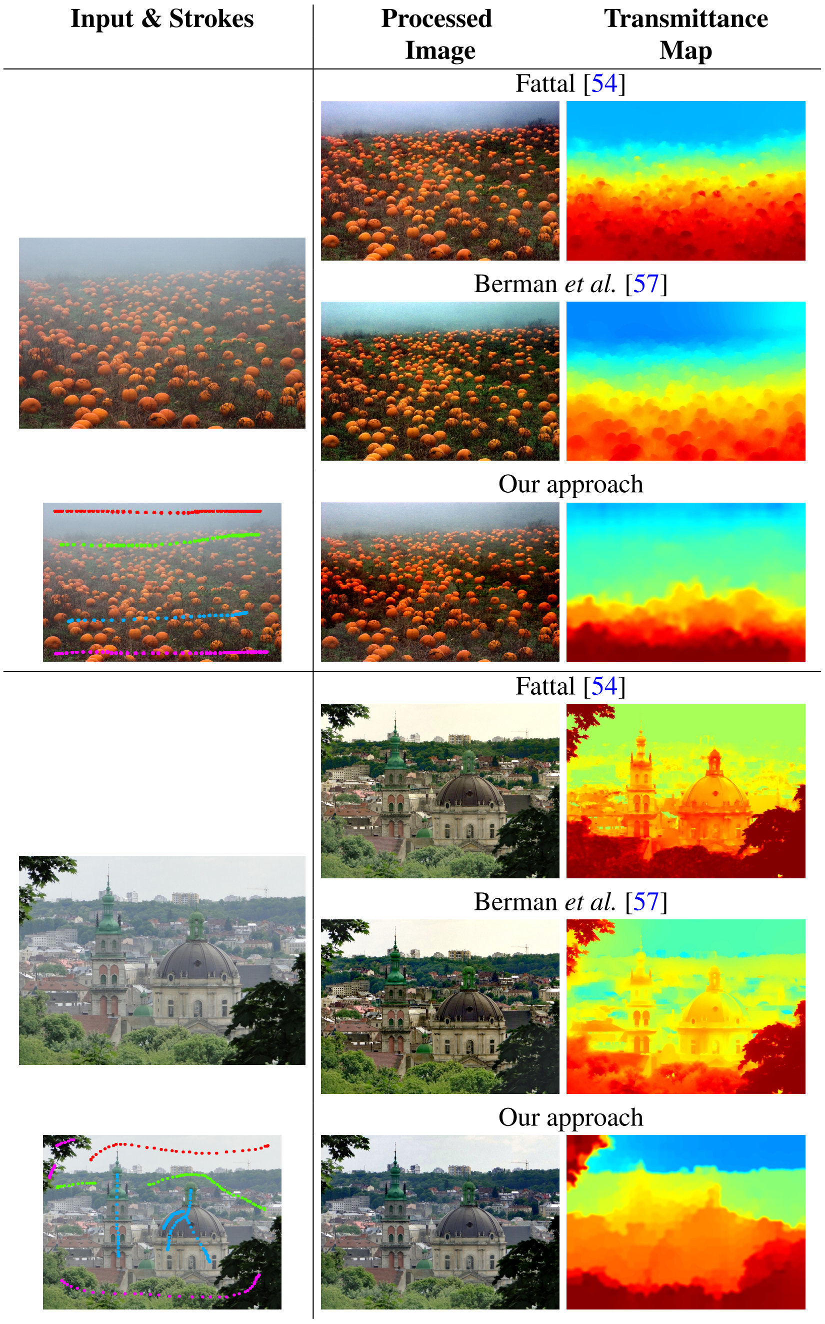


Figure 3.12: Comparison of our results and state-of-the-art single image dehazing methods. Our approach obtains a less fine-grained transmittance estimation but similar dehazing visual results.

3.6.3 Comparison with previous work

An important advantage of our system is that the proposed paradigm is generic, i.e., the same system works for different applications. In this chapter, we have presented three interactive image processing applications that use our proposed system.

This section presents a set of representative visual results from all of them, comparing our solution with well known ad-hoc methods for each application. The following results validate that our generic system can achieve comparable quality in the visual effects than specific ad-hoc methods for each of the applications.

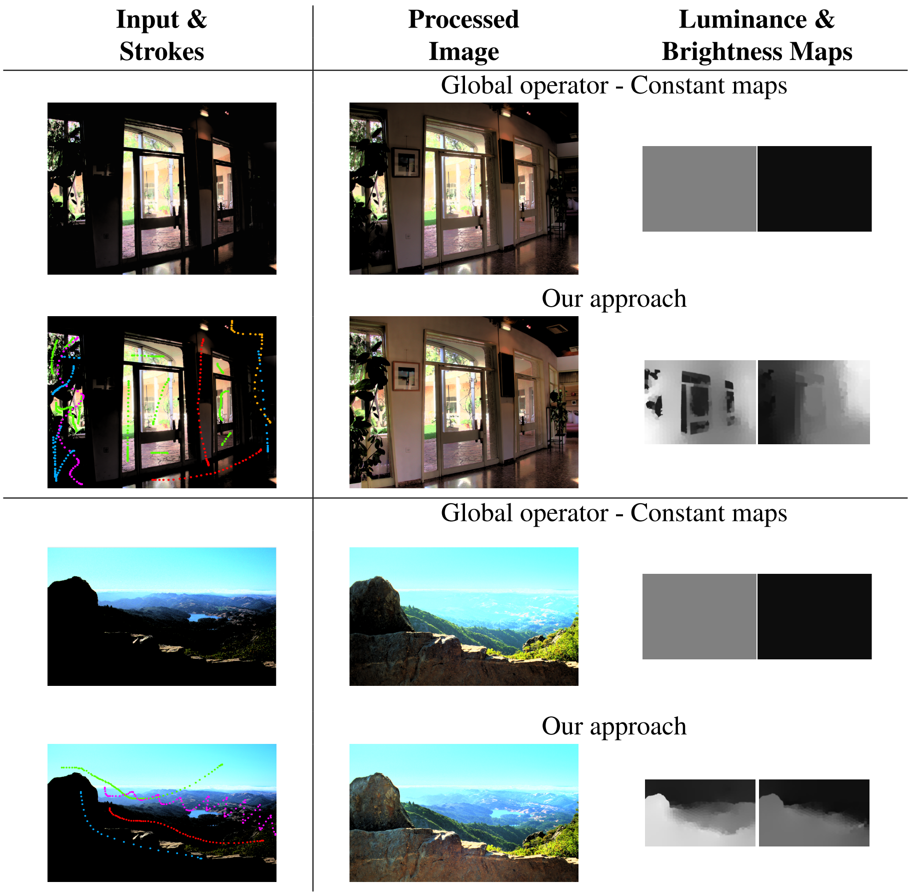

Input & Strokes
Processed Image
Luminance & Brightness Maps
Global operator - Constant maps
Our approach
Global operator - Constant maps
Our approach

Figure 3.13: Comparison of our HDR tone mapping application results. We can see that, with a few user strokes, the brightness and contrast adjusted locally work better than global operators.

Figure 3.11 shows two examples from the results of our depth of field application side by side with the results obtained with a well known approach from Yu *et al.* [52]. Both the depth map and output images are similar, but our approach works with a single image instead of a sequence.

Figure 3.12 compares our results for dehazing against state-of-the-art single image dehazing methods [54, 57]. Note that our transmission maps may look less fine-grained than the other methods, because our propagation system is a superpixel based approach. However, the final visual effect in the dehazed images are of similar quality.

Figure 3.13 compares our results for tone mapping against a global tone mapping operator to adjust the luminance and contrast in the image. Our local tone mapping operators allow the user to obtain better solution than global operators. With a few strokes, the user can point which regions need to be illuminated or darken and adjust each region contrast.

Figure 3.14 shows an example of the results of our HDR tone mapping application and the interactive tool proposed by Lischinski et al. [69]. Both provide the user an intuitive and local control of the region brightness but our method is faster and also provides the adjustment of the image contrast.

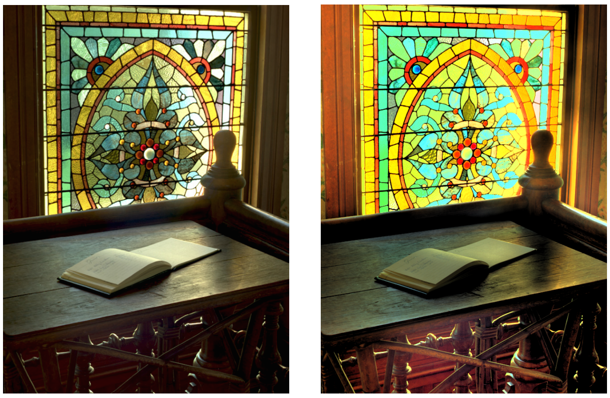


Figure 3.14: Tone mapping produced by (a) Lischinski et al. approach [69] and by (b) our interactive application.

3.7 Conclusions

To demonstrate the suitability of our approach, we have implemented three interactive applications to apply complex well-known filters and effects: depth of field, dehazing and tone mapping. The three applications use the same propagation scheme presented in previous chapter, and their results demonstrate that our generic propagation system achieves comparable image effects than related methods which are ad-hoc for a single specific problem. Besides, thanks our paradigm, these applications can run as interactive tools, with very low computational requirements. This opens the opportunity to bring this type of editing tools to mobile and embedded devices.

We believe our work can inspire both new interactive editing applications, as well as future research on interactive editing tools. Potential lines of future work that can be inspired from the combination of the presented work and previous work are multiview approaches for transferring edits between views [65] or for depth editing and navigation [66]. Besides, instead of plain images, our approach could be extended for material appearance [68] or even material transfer [67] applications, given the adequate adaptation and identification of the involved magnitudes to material space.

Chapter 4

Semantic segmentation models: transfer to other domains

Semantic scene understanding is crucial for autonomous systems to operate in real-world scenarios. Recent deep convolutional neural networks (CNNs) have demonstrated to be an effective approach for semantic image segmentation, especially for tasks where plenty of labeled data is available. However, many applications need to learn new specific classes but do not have a lot of labeled training data. This chapter presents solutions to this problem based on transferring the knowledge from existing CNN models to different classes and domains using two common strategies: 1) fine-tuning a previous model for semantic segmentation; 2) using existing models to obtain image features and use them to represent and classify image regions.

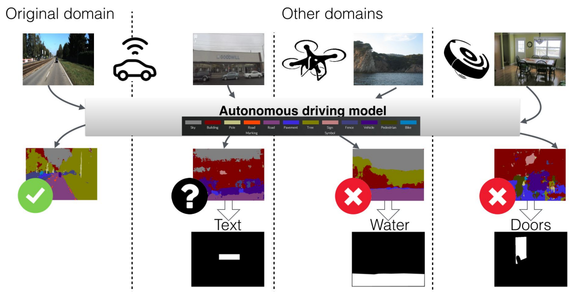


Figure 4.1: Given an existing model, how can we use it to learn new classes, not included initially, which are required for different domains? Many available pre-trained models provide semantic segmentation for images from common scenes such as urban ones (left), however to adapt this to new and more specific tasks (e.g., detect water areas from aerial images), we need to build on top of those models.

4.1 Introduction and Related work

As previously discussed, this thesis contributes on different ways of building systems that use prior human knowledge to obtain a dense labeling of an image. The work in this chapter is focused on supervised learning approaches for semantic segmentation. i.e., differently from the previous chapters, the prior knowledge is not received interactively and incrementally but on a whole set of training labeled data. In particular, we focus on the state-of-the art learning approaches for semantic segmentation, based on deep learning, and situations where the available training data is not enough to train one of those models from scratch. These cases are typically solved through transfer learning and domain adaptation strategies.

Semantic segmentation and deep learning. Semantic image segmentation, or dense image labeling, assigns a category label to each image pixel. This problem has been widely studied in the past and, as many other applications, it has achieved extraordinary results with deep learning based approaches [14, 73–75].

CNNs have become a popular choice because of their effective feature generation and end-to-end training. In order to train a new CNN model from scratch, or to adapt an existing CNN to new applications or inputs, it is required to have large amounts of annotated training data. In many cases, obtaining this training data may be not feasible, so we find many recent works focused on how to deal with this lack of ground truth data, for example by generating photo-realistic imagery as ground truth [76].

Transfer learning strategies. Although successful CNN models are typically shared with the research community, not every concept or object that we need to identify for different applications is already represented in those pre-trained models. Many applications typically need to operate in very specific environments and often happens that the amount of training data for the required visual recognition tasks is insufficient to train a CNN model from scratch. Thus, reusing the existing models by adapting them to the new target domain is worth to pursue and has been shown very successful in the computer vision community. Two major CNN-based strategies for transfer learning are:

1. to fine-tune a pre-trained model with additional data from the new domain [77].
2. to consider intermediate outputs of a pre-trained model as features and train a new classifier with those features extracted on the new domain data [78].

End-to-end semantic segmentation approaches. On one hand, a common approach to adapt pre-trained models is fine-tuning. It consists of training a neural network with the new data and target classes, but initializing part of the network with parameters previously learned for other tasks. Fine-tuning a pre-trained model is a direct way to transfer knowledge learned in one domain to another. For example we find works to learn to transfer image style [79] or to adapt to medical imagery tasks [80]. This approach is effective but not very flexible since it requires data and infrastructure to re-train the CNN model. Although fine-tuning requires much less data than training from scratch, acquiring enough data could still be challenging and the process requires certain infrastructure with GPU computation available.

Superpixel segmentation and classification. On the other hand, CNNs have equipped computer vision systems with the capacity to automatically learn more complex features, compared to typical hand-crafted features used. This approach consists on using an existing model as a feature extractor where the intermediate layers have been observed to correspond to representative features.

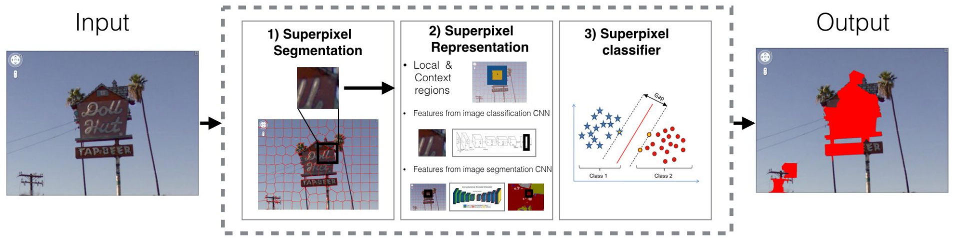
Until the success of convolutional neural networks, most object detectors and recognition tasks were based on sophisticated classifiers that relied on hand-crafted features [81]. Local features have been extensively investigated over the last few decades with the aim of designing descriptors which are distinctive and robust. Convolutional neural networks work like learnable local filters and the outputs of intermediate layers have been observed to correspond to representative features [82]. Recent works demonstrate the features learned when training on ImageNet data for object recognition [83] have been directly applied successfully on other problems such as activity recognition [84], in face recognition [85]. These approaches based on [18], generate region proposals and calculate CNN features there to classified separately each proposal. More recent works, as [86], proposes generate and classify region proposals simultaneously, in order to eliminate the bottleneck of region proposal computation and achieve real-time rates. There are other works which extract features from CNN models and combine them with superpixel segmentation to detect change between pair of images [87] or build a rich superpixel representation features calculating features in different scopes [88].

4.1.1 Contributions

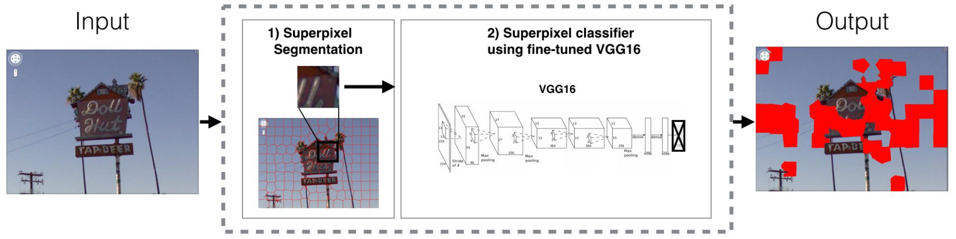
This chapter shows how to apply these two common transfer learning strategies on situations where the amount of data and resources is limited. First, it proposes how to integrate CNN based classification with a superpixel representation considering local and context information. Secondly, it builds and evaluates three different transfer learning pipelines using varied, complementary and realistic scenarios using different public data-sets for different segmentation problems. Finally, it includes two extensions to deal with some of the weaknesses of the end-to-end semantic segmentation approaches.

4.2 Semantic Segmentation pipelines

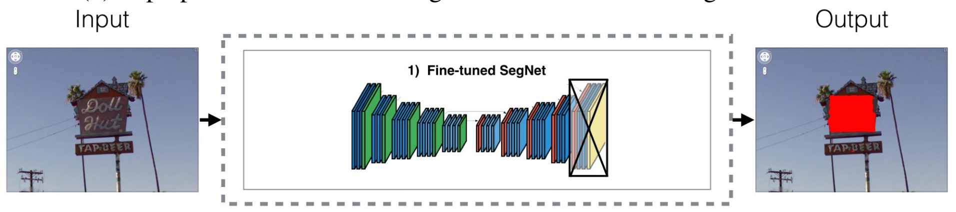
This section describes in detail the two approaches followed to achieved semantic segmentation: 1) image segmentation into superpixels and classification of these superpixels us-


Input
1) Superpixel Segmentation
2) Superpixel Representation
Local & Context regions
Features from image classification CNN
Features from image segmentation CNN
3) Superpixel classifier
Gap
Class 1
Class 2
Output

(a) Superpixel classification using SVM and CNN based features


Input
1) Superpixel Segmentation
2) Superpixel classifier using fine-tuned VGG16
VGG16
Output

(b) Superpixel classification using fine-tuned CNN for image classification



(c) Fine-tuned CNN for image segmentation

Figure 4.2: Transfer learning strategies for semantic segmentation CNN model. (a) Superpixel segmentation and classification using CNN features. (b) End-to-end CNN pipeline for superpixel classification. (c) End-to-end CNN pipeline for per pixel semantic labeling.

ing different features, including hand-crafted or CNN-based features or direct superpixel-patch classification with CNN classification models ; 2) end-to-end system based on fine-tuned CNN segmentation models. The main focus of the pipelines developed and evaluated is to analyze the best way to adapt or re-use information from a existing semantic segmentation models.

4.2.1 Superpixel classification for semantic segmentation

A typical alternative to obtain semantic segmentation of an image is to segment the image in smaller regions and classify each of them. A superpixel classifier based on different superpixel features and descriptors is illustrated in Fig. 4.2 a. Another approach is to design a CNN classification model that directly classifies the patch around each superpixel, such as the pipeline illustrated in Fig. 4.2 b.

4.2.1.1 Superpixel segmentation.

Superpixel segmentation provides a convenient form to compute local image features and it reduces the complexity of image processing tasks. In typical semantic segmentation systems, instead of operating at the pixel level, superpixels are used as the basic unit of a class segmentation. Superpixels are the result of perceptual grouping of pixels, carry more information than pixels and also tend to preserve object boundaries. Superpixels adapt their shape to image content and contours, while typical sliding windows or patches from a regular grid do not (Fig. 4.3). In this pipeline, the superpixels are obtained using the SLIC algorithm [44], but other superpixel approaches could be similarly applied.

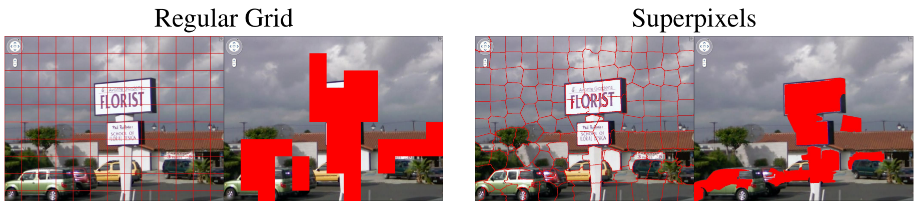


Figure 4.3: Text region classification considering regular grid or superpixel segmentation. As superpixels adapt their shape to object boundaries, they produce better segmentation.

4.2.1.2 Superpixel appearance description.

Given the initial superpixel segmentation, the described superpixel classification pipelines need to represent each superpixel, by computing different descriptors over the pixel values in each superpixel.

Hand-crafted features. Until the success of convolutional neural networks, most object detectors and recognition tasks were based on sophisticated classifiers that relied on hand-crafted features [81]. Some common hand-crafted descriptors used in this work are:

- **Color features.** Low-level features, such as the color, are a common simple choice to represent the superpixel content. Color descriptor can be represented by a 7-bin histogram for the three bands of the LAB color space (21 dimensions).
- **Gradient based features.** Another common way to describe the content of a superpixel is to analyze the distribution of gradient or edge detectors response values over the region. This descriptor can be built using the Structured Edge Detection Toolbox [89]. To obtain the edge probability distribution on a set of pixels, it can

combine the mean, the standard deviation and the Skewness and Kurtosis of the edge probability distribution as an edge descriptor. Histogram of oriented gradients (HOG) descriptor [90] is also a typical edge descriptor which usually is built from a 9 bins for each cell, with a cell size of 8x8 and a block size of 16x16 (2916 dimensions).

- **Semantic features.** The amount of contextual information present in a superpixel can be estimated by examining the average number of object categories and instances per superpixel. A semantic description could be built from any automatic and dense semantic segmentation result of an image as a histogram of the distribution of the pre-trained semantic labels within certain pixel region.

CNN based features. Recent works shows that CNN based features learned at intermediate layers from CNN models are very discriminative for many recognition tasks. Sec. 4.4.1 evaluates the semantic segmentation results achieved with CNN based features compared to more traditional hand-crafted features.

We group the CNN based superpixel descriptors studied according to the actual goal of the original CNN model:

- Classification CNN models. This kind of CNN have been directly applied successfully on other vision problems, including image segmentation [91]. The three well known image classification models consider are: *Alexnet* [83] and *Hybrid-CNN* [92], which share the architecture but use different set of training data and classes, and *VGG* model [93], which is a larger architecture. The descriptor selected is the weights from last fully connected layer (fc_7), resulting in 4096 descriptor dimensions, from each of the models studied.

- Segmentation CNN models. The output of this kind of CNN is trained to directly perform image segmentation, i.e. classify each pixel with a semantic label. The pixel-wise segmentation obtained is often noisy and imprecise in the object boundaries, but provides a valuable coarse segmentation of the main elements of the scene. The output of segmentation models can be used to build a semantic descriptor as a histogram of the distribution of semantic pixel labels within the superpixel region (a bin for every semantic label). The semantic labels considered have been obtained with the public SegNet model [9] but this description could be built from any automatic and dense semantic segmentation result of an image.

4.2.1.3 Superpixel contextual description.

The previous appearance descriptors can be computed in two superpixel scopes, as are illustrated in Fig. 4.4. While *Local* scope corresponds to information of pixels within each superpixel; *Context* scope corresponds to information from the pixels of neighboring. Section 4.4.2 includes a more detailed analysis of the effects of representing the contextual information around a superpixel.

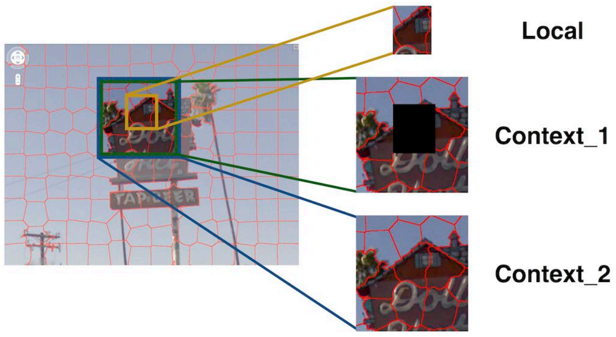


Figure 4.4: Superpixel scopes

However, thinking of common configurations for most of semantic classes in a natural scene, the objects are usually related with their surrounded objects. Road regions occur often at the bottom part of an image, typically touching building regions at the left or right sides. Building regions are often surrounded by sky regions on the upper part, and car regions are often surrounded by road regions. This motivates a simple and compact semantic context superpixel descriptor, which considers the semantic labels of all pixels from its adjacent superpixels (considering 8 pixel neighborhoods).The adjacent pixels are distributed in four separate sets, attending to their position with respect to the centroid of the superpixel: top, bottom, left and right. The distribution of pixel semantic labels in each of those sets, is combined in four histograms. Fig. 4.5 represents how this process works for one superpixel. then, this proposed superpixel context descriptor consist of a 48 bins histograms.

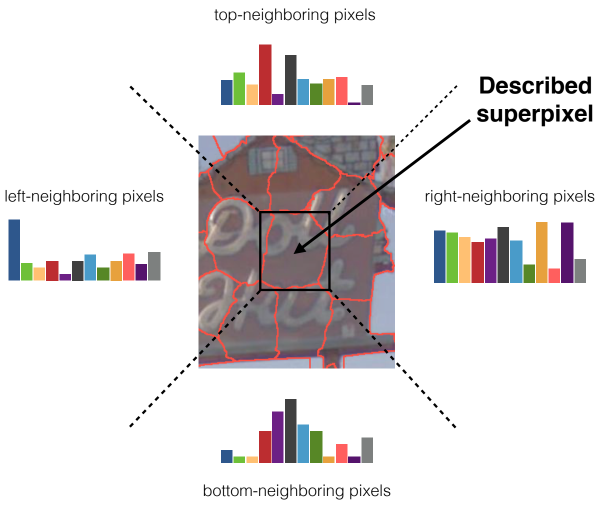


Figure 4.5: Semantic context feature. The distribution of semantic labels of surrounding superpixels is used to build four histograms.

4.2.1.4 Superpixel classification.

Superpixel descriptor based classification. The goal of a superpixel classifier is to identify superpixels from the target class given a set of superpixel descriptors, i.e. to separate superpixels from the new target class from the rest. The classifier learns from positives and negatives examples. Each training superpixel feature vector is stored as a row of training matrix X, and each superpixel ground truth label, is stored in the column matrix Y:

$$X_{(n,f)} = \begin{bmatrix} features(sp_i) \\ \vdots \\ features(sp_n) \end{bmatrix} \quad Y_{(n,1)} = \begin{bmatrix} label(sp_i) \\ \vdots \\ label(sp_n) \end{bmatrix} \tag{4.1}$$

This section proposes using a standard Support Vector Machine (SVM) [94], which solves the classification problem as:

$$\begin{array}{cc} \underset{\alpha}{min} & \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ \text{subject to} & y^T\alpha = 0, \\ & 0 \leq \alpha_i \geq C, i = 1, ..., n \end{array} \tag{4.2}$$

where e is a vector with ones, Q is an lxl positive semidefinite matrix, and $Q_{i,j} \equiv y_i y_j K(x_i, x_j)$, and $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$ are the kernel functions where $\phi(x_i)$ maps x_i into a higher-dimensional space.

Therefore, we have n training examples:

$$\{x_i, y_i\}, i \in (1, n),$$

where n is the number of labeled superpixels, x_i is a vector in the feature space and y_i denotes the class index, taking a value $\in \{1, -1\}$. Our superpixel training set consists of positive and negative superpixel examples of the new class. Figure 4.6 shows some examples of superpixels used to train one a text region segmentation algorithm.

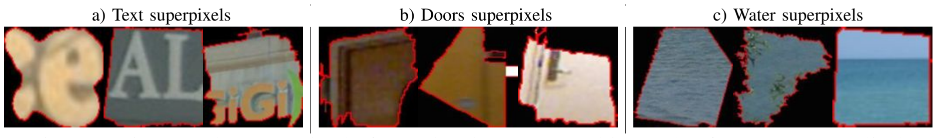


Figure 4.6: Sample superpixels from the datasets used in each of use cases studied.

Superpixel patch classification. A common solution to adapt deep learning models to a new domain is to fine-tune an existing model. This pipeline is illustrated in Fig. 4.2 b, and it can be understand as an end-to-end patch classification. In this pipeline, we fine-tune the publicly available *VGG-16* model [93], which was originally trained to classify images into the 1000 ImageNet object categories [95]. Following a similar idea than R-CNN framework [18], we classify each superpixel separately and build the final image semantic segmentation concatenating the fine-tuned network outputs. Note that, this pipeline is similar than previous superpixel pipeline, where each superpixel is classified individually into a class.

4.2.2 Semantic segmentation with fine-tuned CNN models

As previously mentioned, a common solution to adapt deep learning models to a new domain is to fine-tune an existing model. In this section, we propose how to adapt existing CNN models for end-to-end semantic segmentation. We follow a common procedure to fine-tune a CNN model [79, 80]. First, we replace the last CNN classification layer with a new classification layer. The output of this new layer corresponds with the desired number of outputs for our binary segmentation task. Next, we train the adapted network structure with labeled data from the new target class, initializing this process with the original network parameters. The back-propagation algorithm run during training optimizes the network parameters using Stochastic Gradient Descent algorithm (SGD). The two different pipelines we have built based on fine-tuned existing CNN models are detailed next.

This work considers fine-tuning recent CNN-based semantic segmentation models, such as SegNet [9] or DenseNet [96]. Note that these models are an end-to-end pipeline, where the input is a whole image and the output is the final per-pixel labeling. As illustrated in Fig. 4.2 c), we fine-tune this model to perform a binary segmentation of a new target class. Following standard fine-tuning ideas, we slightly modify the network design to achieve the desired output.

4.3 Experimental setup.

To compare the two common transfer learning approaches (described in previous section), for the particular problem of semantic segmentation, both approaches are evaluated on three varied binary segmentation use cases using public datasets. The experiments consist of training and evaluating a new binary classifier for a new target semantic class when the new target dataset availability is not large enough to train new models completely.

4.3.1 Datasets

This section runs three different experiments to learn a new semantic class. These three new target classes were chosen among classes where we could find enough labeled data and were not already included on the reference semantic segmentation models. Besides, they cover a variety of use cases, since they span different levels of man-made characteristics and different levels of difficulty due to the variety of samples.

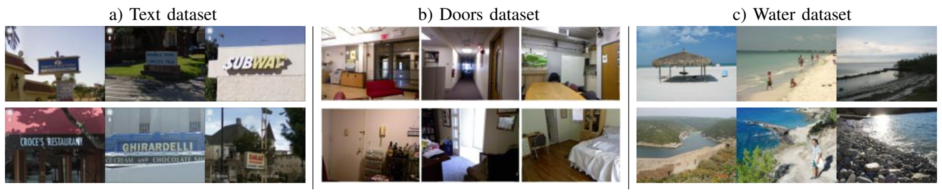


Figure 4.7: Sample images from the datasets used in each of use cases studied.

Text segmentation

Text regions in natural images present a huge variability in appearance and layout, but they possess distinguishable and discriminative properties. The data used in this experiment is from the *Street View Text* dataset [97], where most of the images come from business signage and exhibit a high degree of variability in appearance and resolution. Figure 4.7 a) shows a few representative images from this dataset. The provided training set contains 100 images, while the testing set contains 249 images. In average 3.38% pixels per image are labeled as Text.

Doors segmentation

Doors are important landmarks indoors, and automatic visual door recognition has major difficulties due to the variety and diversity of appearance and shapes of door regions. The RGB data used in this experiment is from the *NYU v2* dataset [98] (Figure 4.7 b). In this experiment, only the images which have pixels labeled as door are considered. The resultant training and testing sets contain 239 and 165 images respectively. In average 7.63% pixels per image are labeled as Door.

Water segmentation

Detection of image regions containing water is a challenging problem, since they do not have a defined shape and the appearance suffers strong changes do to lighting, reflection, etc. The data used in this experiment is from the *LabelME* dataset [99] (Figure 4.7 c). As positive Water class examples, regions from all labels related to it are used: sea, water, lake, pond and ocean; and the rest of labels are considered as no-water. The training and testing sets contain 100 and 194 images randomly picked from these collections: *static waterfront boston outdoor july 2005, web static park garden outdoor, static web outdoor spain, static coast palafrugel outdoor spain*. In average 28.16% of the image pixels from the images used from these collections are labeled as Water.

4.3.2 Training details

A standard SVM implementation in OpenCV, which is based on LibSVM library [100], is used as the *superpixel classifier*. In all the experiments, positive training samples the superpixels contain more than 50% of their pixels labeled as the target class. Note that superpixel segmentation solution can often provide superpixels where not all pixels belong to the same class. Then, the number of positive training samples for each classification task is different: 570, 1066 and 5766 positive examples to Text, Door and Water segmentation respectively. As negative examples, superpixels are randomly chosen from images in the training set with zero pixels labeled as the target class.

For both the extraction of CNN features and the *fine-tuning* of CNN models, the open source Deep Learning library Caffe [101] is used. All fine-tuning of the modified version of the networks have run around 60,000 iterations, using a NVIDIA GeForce GTX TITAN X GPU with 12GB RAM.

4.3.3 Evaluation metric.

For all the experiment results, the evaluation metrics are calculated per pixel. The average Precision and Recall from pixel classification are computed using the training/testing set splits provided in the corresponding public datasets. The average Precision P and Recall R for all images are computed as:

$$P = \frac{TP}{TP+FP} \quad R = \frac{TP}{TP+FN} \tag{4.3}$$

where True Positives TP are pixels correctly classified as *target* class, False positive FP are pixels misclassified as *target* class and False Negatives FN are pixels misclassified as *No-target* class.

4.4 Experiments Superpixel classification

This section compares the effectiveness of the different ways to describe superpixels, presented in Sec. 4.2.1.2 and Sec. 4.2.1.3 , to classify them into new semantic classes.

4.4.1 Superpixel appearance representation

These first experiments are focused on the problem of text segmentation in natural images which compare typical hand-crafted and CNN-based superpixel features. It evaluates the performance obtained with a superpixel classifier trained separately with each of the superpixel descriptors considered using the train/test split provided in the SVT dataset.

The average results for all test images are shown in the Table 4.1 and Fig. 4.8 shows an text segmentation example. Common simple superpixel representations, such as color C or information about edges within the region (E or Ho), do not provide any discriminative

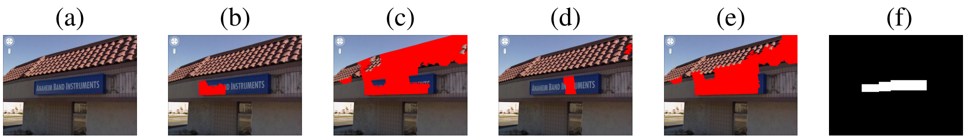


Figure 4.8: Text region segmentation (in red) using different features. (a) Original image. (b) Hand-crafted features. (c) CNN-based features. (d) Semantic features. (e) Previous features (b,c,d). (f) Ground truth.

power for this task. The same occurs with the semantic-local descriptor. The performance of the CNN features (A and H) are similar, possibly due to they share the network architecture. They obtain the highest amount of text in the dataset, around the 0.55. Combining descriptors ($Low1$, A and S), the Recall of CNN features is not improved. It is due to the hand-crafted features are already encapsulated in the CNN descriptor. Then, this results shows how CNN models capture reasonably well superpixel information, independently of use CNN model.

Table 4.1: Text/No-text classification results. Precision P and Recall R are calculated per pixel and %T is the percentage of the image detected as Text.

Local features	Superpixel		
	Local		
	P	R	%T
Hand-crafted			
Color (C)	0.17	0.07	1.95
Edges (E)	0.35	0.01	0.05
Hog (Ho)	0.30	0.01	0.06
Low1 ($C+E$)	0.26	0.13	1.40
Low2 ($C+E+Ho$)	0.30	0.01	0.06
CNN features			
AlexNet (A)	0.31	0.55	5.76
HybridCNN (H)	0.33	0.56	5.61
Semantic			
Semantic (S)	0.28	0.03	3.14
$Low1+S+A$	0.32	0.57	5.49

4.4.2 Superpixel context representation

Superpixel feature representation is calculated in two different scopes: from the superpixel itself to a region around it. Each representation is analyzed training various superpixel classifiers using the same CNN based features but varying the region in which these features have been calculated. This experiment is focused on different ways to represent the contextual information around a superpixel. Given a superpixel, the context region is defined as the region encloses all pixels from its adjacent superpixels. The CNN features are computed into the context region excluding or not the superpixel itself, i.e., $context_1$ and $context_2$ regions respectively (Fig. 4.4).

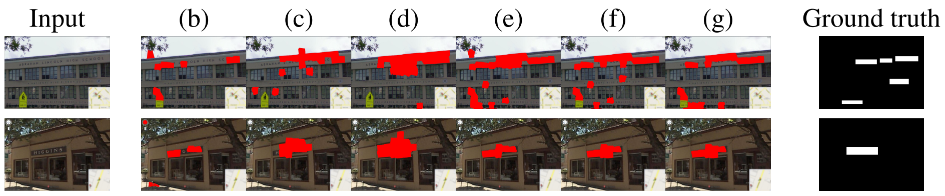


Figure 4.9: Different superpixel context representation (Text classification task): (b) $local$, (c) $context_1$, (d) $context_2$, (e) $local$ + $context_1$, (f) $local$ + $context_2$, (g) $local$ + $context_1$ + $context_2$.

Each superpixel classifier has been tested in the Text classification problem. The results are summarized in Fig. 4.2 and Fig. 4.9. Based on these results, including context features ($context_1$ or $context_2$) help to detect more Text regions (*Recall*) and with less false-positives regions (*Precision*), independently that context region considered. Both context descriptors obtain similar results, then, they could be used indistinctively. In next experiments, $context_1$ is used to represent the superpixel context.

Table 4.2: Superpixel context representation in Text classification.

Scopes	Precision	Recall
$local$	0.34	0.70
$context_1$	0.23	0.59
$context_2$	0.27	0.68
$local$ + $context_1$	**0.35**	**0.76**
$local$ + $context_2$	**0.34**	**0.77**
$local$ + $context_1$ + $context_2$	0.37	0.74

4.5 Performance of transfer learning pipelines

This section evaluates the two studied transfer learning strategies to achieve the semantic segmentation of a new target class in the three different applications described: Text, Doors and Water region segmentation.

Fig. 4.10 shows examples of each applications, which achieve different performance. Appendix B shows additional examples. As it could be expected, Door classification is a

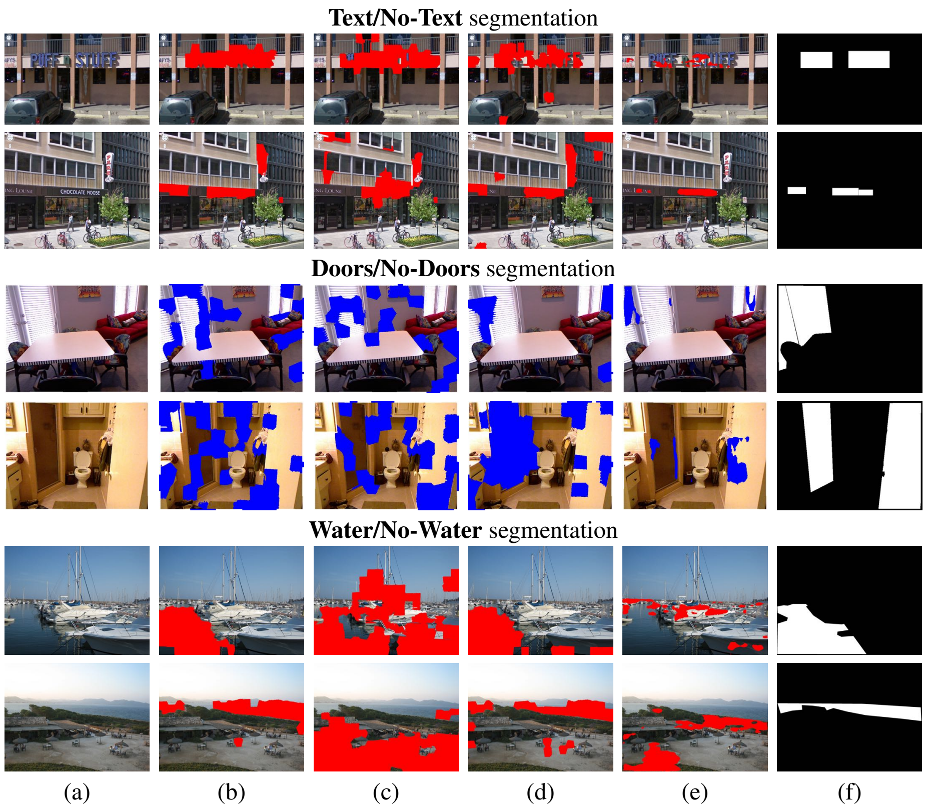

Text/No-Text segmentation
Doors/No-Doors segmentation
Water/No-Water segmentation
(a)
(b)
(c)
(d)
(e)
(f)

Figure 4.10: Representative examples of pixel segmentation (in red or blue) using different transfer learning pipelines. (a) Original image. (b) CNN-based features. (c) Semantic features. (d) Fine-tuned CNN model for image classification and (e) for image segmentation. (f) Ground truth.

more challenging tasks, due to large variations on appearance and shape of door regions, as well as being hard to discriminate from other common indoor surfaces, such as furniture. As we can observe in Fig. 4.7 the set of training data available is not enough to cover all the variability of this class, while the other experiments intra-class variation (water and text) seems to be better captured by the training set, even though we have similar amount of images for each of them. Note that the ground truth images contain some errors as shown in Fig 4.10, e.g., the second row shows ground truth with actual text regions unlabeled.

Table 4.3 shows the average Precision and Recall obtained for each of the three ex-

Table 4.3: Precision P and Recall R of semantic pixel segmentation for each application (Text, Door, Water) using different transfer learning pipelines. Figure 4.10 shows examples of the best ones.

Pipelines			Text		Door		Water	
			P	R	P	R	P	R
CNN based SUPERPIXEL FEATURES + SVM superpixel classifier								
	Feature	size						
	AlexNet_fc7 (A)	8192	0.33	0.65	0.13	0.44	0.90	0.85
	HybridCNN_fc7 (H)	8192	0.34	0.70	0.12	0.40	0.87	0.86
Fig 4.10(b)	VGG_fc7 (V)	8192	**0.35**	**0.76**	0.12	0.44	0.88	0.85
Fig 4.10(c)	SegNet_hist (S)	60	0.19	0.43	0.11	0.30	0.66	0.81
	Combined							
	$V+S$	8252	0.32	0.67	0.11	0.44	0.90	0.69
Superpixel segmentation + FINE-TUNING of CNN for image classification								
Fig 4.10(d)	fine-tuned_VGG		**0.20**	**0.77**	**0.20**	**0.44**	**0.88**	**0.87**
FINE-TUNING of CNN for image segmentation								
Fig 4.10(e)	fine-tuned_SegNet		**0.53**	**0.58**	**0.28**	**0.10**	**0.58**	**0.27**

periments (Text, Doors, Water). Each row corresponds to a different configuration of the semantic segmentation approach (Sec. 4.2). All descriptors are calculated on the two scopes: *Local* and *Context* (in particular using $context_1$) which has been demonstrated in previous section 4.4.2 that using both scopes significantly improves performance.

While recall is always higher on the superpixel based pipelines (i.e., we were able to correctly label more image pixels from the target semantic class), the accuracy results point to two different conclusions. On one hand, in *Text* classification, the *fine-tuned_SegNet* end-to-end classifier, achieved the best precision. This can be explained because the *Text* classification problem is the closest domain to the type of images that were used to train SegNet model (urban scenes, including a class for signs, which often contained text-like areas). On the other hand, in *Water* and *Text* classification, the superpixel-based classifiers obtain significantly higher recall (and comparable precision) than the fine-tuned model for image segmentation.

4.5.1 Different CNN based superpixel representations

The first five rows in Table 4.3 show Precision and Recall of the semantic classification of pixels using different CNN superpixel representations and SVM classifier. Given a superpixel patch, rows A, H and V show results using features extracted from layers of different CNN models for image classification, while S descriptor is obtained as the histogram of the output within the superpixel patch from a CNN model for image segmentation. There are several common points for all these experiments: combining the two types of CNN based descriptors ($V+S$) does not improve the Recall of the best of them separately. This confirms the amount of information enclosed on S is definitely more compressed but less detailed, something that could be expected from the very different size of descriptors (column *size* in the table).

If we compare both types of superpixel based classification pipelines (the best SVM based superpixel classifier, row V, and the fine-tuned model for image classification, *fine-tuned_VGG*), is not surprising that they get to similar quality on the results. They both take into account superpixel restrictions and use features learned in previous CNN trained for image classification. The difference is how they train given this information. The first case trains an SVM with this information, while the second fine-tunes the existing CNN, i.e., trains the last layers to adapt them to the new domain.

Therefore, using CNN features extracted by last CNN layer is equivalent to fine-tune the CNN model in terms of accuracy of the results (only in *Text* experiment, the fine-tuned model *fine-tuned_VGG* obtains worse results). The main difference between these pipelines is during the training stage. While a simple SVM can be trained on the CPU of a common desktop computer and it just takes the order of minutes, fine-tuning a CNN model in the same conditions can take several days. At the end of this section, we present a more detailed analysis of which pipelines are more suitable when we have time or resources restrictions.

4.5.2 Superpixel-based classifiers vs end-to-end segmentation.

As previously mentioned, if we compare the results obtained by superpixel-based classifiers (V and *fine-tuned_VGG*) and the end-to-end pipeline obtained with the fine-tuned model for image segmentation (*fine-tuned_SegNet*), we can observe more significant differences in the performance obtained, which highlight the influence of the following aspects, which should be taken into account when targeting a new semantic class segmentation:

Different domains. As previously mentioned, the different behavior of the *Text* experiment (achieving the best precision using the fine-tuned SegNet to directly obtain the binary image segmentation) is due to the fact that this target class is closer to the domain considered in the original label set. So in general, we could suggest to use a superpixel-based classifier for the transfer learning pipeline when the new target class is from a very different domain than the data from the existing CNN model.

Amount of training data. Our use cases are motivated by not having enough labeled data to train a model from scratch. A benefit we have observed in the superpixel based approaches is that with the same amount of labeled training images, we actually have a larger set of training samples for superpixel based approaches. This provides better models and results with limited amount of training data.

Limited resources. Another important aspect to study when choosing the most convenient way to implement a new classifier based on existing models is the type of resources we have available (time restriction or availability of GPU), and how they affect our target application or platform, during training or prediction stages.

Regarding the **training** stage, we should note that the *SVM based superpixel classifier* has much lower computational requirement to train a new binary segmentation model, because we only run CNNs forward steps to obtain the features, which is relatively fast, even without using GPU (around 5s to get features for a 50 image batch on a CPU). Then, the SVM training is a fast step, e.g., training a superpixel classifier with 15766 examples using a 4096-dimensional descriptor for each sample, takes around 4 minutes in a common desktop computer. However, we estimate that fine-tuning a CNN model could take around 30 hours executing 60.000 iterations in similar conditions (CPU). For this option to become more affordable (as we have done in our experiments) we should fine-tune the models using a GPU. Therefore, the SVM based pipeline is the best option without a GPU available for the training stage, since using a CNN only to extract features is not very demanding operation that can be run without a GPU.

Regarding the **prediction** stage, i.e., to achieve the semantic segmentation of a new image, we should note that there are smaller execution time differences among the different pipelines. This is due mainly because we always use a CNN model, either to extract the features or to directly predict the final segmentation. The execution time in the different pipelines is summarized in (4.4), where T_a is the time for a superpixel based SVM classifier, T_b is a superpixel based CNN image classifier, and T_c is an end-to-end CNN image segmentation pipeline:

$$\begin{aligned} T_a &= T_{sp} + |sp| \cdot (T_{cnn} + T_{svm} + T_{out}) \\ T_b &= T_{sp} + |sp| \cdot (T_{cnn} + T_{out}) \\ T_c &= T_{cnn} \end{aligned} \tag{4.4}$$

T_{sp} is the time to segment an image into $|sp|$ superpixels, T_{cnn} is the time to run the a forward pass of an image through a CNN, T_{svm} is the time to classify an superpixel with an SVM and T_{out} is the time to build the output image combining individual superpixel classification. Note that, T_{svm} and T_{out} can be ignored, since their times are despicable compared to the other steps. Besides, $|sp| \cdot T_{cnn}$ is almost equivalent to T_{cnn} if we process the superpixel patches in batches (note that CNN evaluation can process batches of images in a parallel manner), so the main difference between T_a, T_b and T_c is the superpixel extraction time (6s per image in the algorithm we use). An important aspect to consider about the requirements during prediction stage is that often the deployment of trained models for the final applications (e.g., deploy the learned model on a robot on-board computer), has stronger time and resources limitations. Therefore, the end-to-end pipeline is the best option for a resource limited platform for the *prediction* stage, since as we have seen, the training is performed beforehand, and the only step needed for prediction is a task that can be performed reasonably fast without a GPU.

4.6 Additional improvements to semantic segmentation models

Looking into the results from previous section, we observe how the different presented semantic segmentation pipelines provide good results for certain applications, such as the *Text* and *Water* segmentation examples, while other tasks, such as *Doors* segmentation, do not achieve acceptable results. This section explores additional steps to improve semantic segmentation results in such tasks. Usually, this kind of poor results are due to poor training data, e.g., there are few training examples, which do not cover the variability of the modelled class, or the quality or detail of the available examples is not good enough. We explore two strategies to improve semantic segmentation pipelines from two different points of view, as summarized in Fig. 4.11:

- Data pre-processing: augmentation of sparse training data into dense training data. Sometimes the available training data is not detailed enough to train certain types of models, for example to train semantic segmentation models, we typically need dense labeling of the training data. We show how to automatically augment sparse labeled data in order to enable the training of a semantic segmentation model as those studied in previous sections. This is detailed in Section 4.6.1.

- Data post-processing: we combine semantic segmentation results with geometry-based information. Given a semantic segmentation, the goal is to refine the obtained predictions near the object boundaries where usually the predictions are wrong. This is detailed in Section 4.6.2.

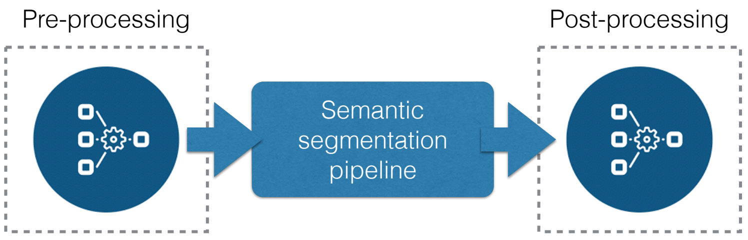


Figure 4.11: Standard semantic segmentation pipelines can be improved with additional modules to pre-process the input data or post-process the output to refine the results.

4.6.1 Pre-processing input data

One of the biggest issues to train or fine-tune existing CNN semantic segmentation models is that they require example training data. However, there are many domains where obtaining large amounts of good quality dense labeled segmentation data, which is required to train such approaches, is highly costly and tedious to obtain. For example, tasks to monitor different aspects of wildlife can highly benefit of automatic semantic segmentation approaches, from animal recognition in videos [102] to coral identification in underwater survey imagery [103]. Unfortunately, datasets of this kind often only provide a weakly labeled ground truth. In this section we propose how to augment the ground truth for such cases. In particular, we present an approach for a coral segmentation application, a use case that would highly benefit from automated monitoring, but that lacks large amounts of quality and accurately labeled data. Our system presents promising results, and is able to effectively learn coral segmentation and provides a flexible end-to-end pipeline.

4.6.1.1 Augmentation of sparse training data

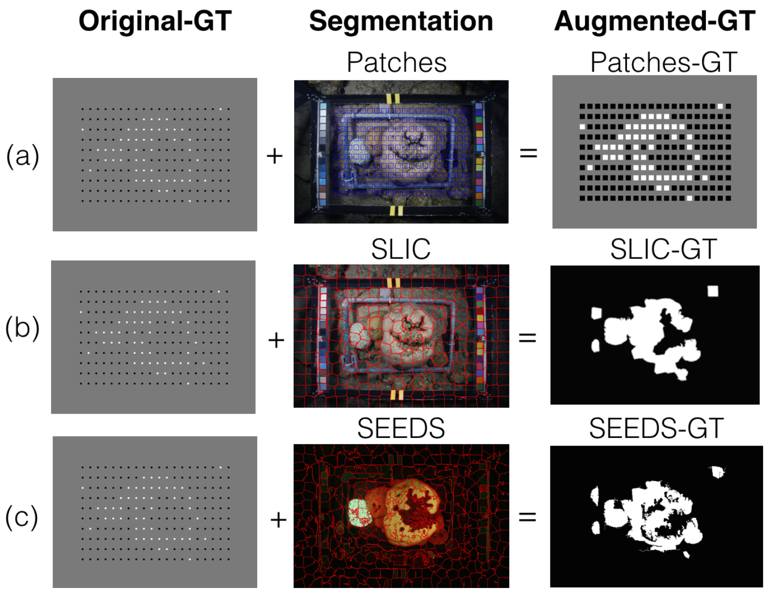


Figure 4.12: Ground truth augmentation methods that we considered. (a) small patches around original-GT labeled pixels; (b) SLIC and (c) SEEDS superpixels, computed on RGB or fluorescence images, used to expand the original-GT. SLIC and SEEDs can be augmented using either RGB or fluorescence image superpixels but fluorescence yields a much better segmentation.

The main challenge considered in this section is how to learn a good semantic image segmentation given a very sparse ground truth to learn the model. Typically to train a CNN for semantic segmentation dense ground truth is needed. We evaluate three strategies to obtain this dense labeling, as shown in Fig. 4.12.

Patches-GT. This strategy is the more straightforward. We expand the labeled ground truth pixels into labeled patches around those pixels. This strategy assumes that the surrounding pixels of a labeled one are the same kind. Several patch sizes were tested and 25x25 pixel patches gave the best results (using 1078 x 976 images) providing 125000 labeled pixels per image instead of 200.

Superpixels (SLIC-GT, SEEDS-GT). We apply these superpixel segmentation methods to the images. This allows us to match the original labeled pixels to each segmentation. This method gives a better and more accurate solution. The outcome augmentations of SLIC [44] and SEEDS [104] superpixels are similar. Visually, SEEDS-GT fits better to the shape of the coral. These methods can fail specially when the corals are too small or the have holes. Nevertheless, these approaches seem pretty similar to the RGB images. This superpixel augmentation can be obtained from any of the multi-modal images (see Fig. 4.12), and is independent from the image channel used in the final segmentation step. The experimental results, from the next section, analyze the differences of using with different augmented ground truths in our pipeline.

4.6.1.2 Results

The final step consists of training the model with the augmented-GT. The state-of-the-art image segmentation systems use CNN based models, which offer excellent accuracy. Our goal is to adapt existing semantic segmentation models to our target classes. The following experiments analyze different aspects and variations of our approach for coral segmentation and compare the results obtained with prior work on the same data.

Data-set. All the following experiments are run on the Eilat Fluorescence Corals dataset [103]. The dataset consists of 212 coral annotated multimodal image-pairs: RGB and fluorescence images. There are 200 labeled pixels per image, assigning to each of them a label from coral and non-coral classes[1]. Note that this ground truth is very sparse, since images have 1078 x 976 resolution. The data is split into a training-set of 142 randomly selected image-pairs, and a test-set with the remaining 70 image-pairs.

Evaluation. We use standard accuracy, recall and precision scores for the evaluation of the results computed according to different strategies:

Original-GT based sparse scores. The scores computed based on the original ground truth (Original-GT) are not fully representative, as it will be shown next. Intuitively, 200 pixels labeled out of around a million per image are not a dense ground truth for dense image labeling.

Superpixel-GT and Manual-GT based dense scores. The augmented ground truth we generate based on superpixels (Superpixel-GT) is an approximated but dense labeling, which as shown next gives a reliable evaluation. The fact of having very sparse ground

[1]http://datadryad.org/resource/doi:10.5061/dryad.t4362

Table 4.4: Coral segmentation (average pixel classification accuracy). Training and evaluation with different ground truth (GT).

Evaluation: *Training*:	Original-GT **(sparse)**	Patches-GT	SLIC-GT **(dense)**	SEEDS-GT **(dense)**
Original-GT	0.56	0.53	0.43	0.42
Patches-GT	0.77	0.80	0.67	0.67
SLIC-GT	**0.81**	0.80	**0.89**	**0.90**
SEEDS-GT	0.78	0.77	0.85	0.86

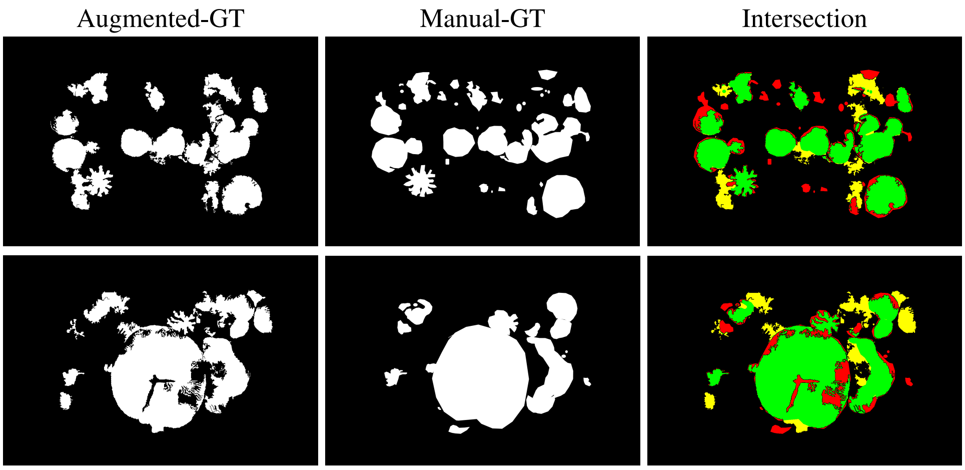


Figure 4.13: Examples of Augmented-GT and Manual-GT. The intersection of both shows green/red/yellow pixels when labeled as coral in both/only manual-GT/only augmented-GT respectively.

truth is a challenge not only to train but also to evaluate in a meaningful way the dense labeling results. The representativity of this augmented ground truth can be seen in multiple visual results. Besides, we include comparisons using a few (7% of the testing data) detailed manual segmentations (Manual-GT) performed by an expert. This helps to further validate the Augmented-GT and the segmentation results. The average accuracy of the values in the augmented-GT with respect to the Manual-GT is of 93% (for the 5 images with Manual-GT available). Fig. 4.13 shows examples comparing these two segmentations.

4.6.2 Post-processing semantic segmentation

As mentioned before, other possible path to improve semantic segmentation algorithms is to refine the final output. Automatic dense semantic estimations are usually noisy in the the object boundaries. We consider some simple image post-processing techniques which may be suitable to improve the results. Depending on the application domain, it is possible to make geometric assumptions, which help us refine the final segmentation results.

In this section we describe an strategy to eliminate false positive and improve the robustness of a CNN-based semantic segmentation approach by integrating simple prior knowledge about geometric information with the CNN output.

4.6.2.1 Combining geometric info with CNN-segmentation output

For example, problems as Door classification, it is possible adding a light geometry based post-processing step which helps detecting regions of interest in the image. In particular, a sign detector based on generic rectangular hypothesis generation, presented in [23], can be used to create rectangular hypotheses over all the image. Then, each rectangular sign candidate is filtered and classified into Door or No-Door according of the previous semantic segmentation obtained.

The process to create rectangular hypotheses starts with a search for possible rectangular in the image. Since there no restrictions on the image to be processed, this approach detects straight segments and cros points (corners) between them all over the image. According to the relative position of the components of each cross point, we find four different types of corners, as shown in Fig. 4.14.

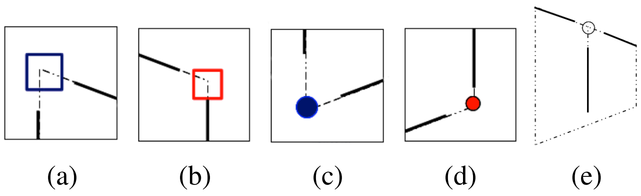


Figure 4.14: Types of corners detected: (a) Top-left, (b) Top-right, (c) Bottom-left ,(d) Bottom-right. (e) Multiple-type corner: TL+TR.

To generate the rectangular regions, corners are grouped with each other trying to get sets made of as many compatible corners as possible. Then, we obtain compatible sets made out of one, two, three or four image corners. All of these sets, except the four-corner ones, need a post-processing to estimate the complete shape of the rectangular region, as shown in Fig. 4.15.

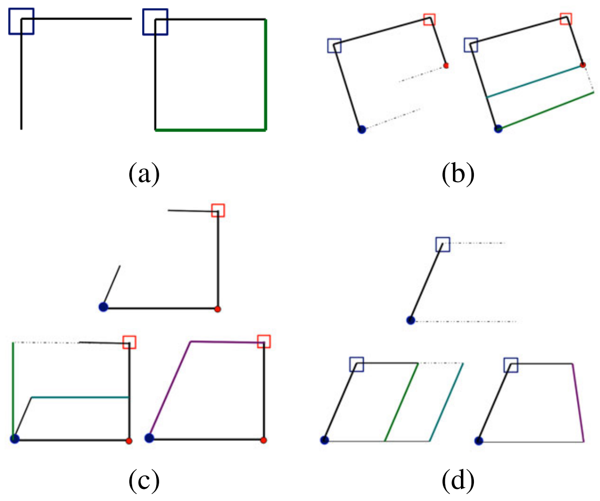


Figure 4.15: Rectangular hypothesis generated from singles corners (a) and from sets of four (b), three (c) and (d) corners

4.6.2.2 Results

Fig.C.3 shows examples of the improved Door segmentation. Appendix C includes additional results. Table 4.5 summarizes the results obtained with a subset of the previous Door detection problem described in previous Section 4.3. In particular, we only select a sub-set of the provided dataset.

Table 4.5: Precision and Recall (per pixel) for door segmentation, combining a rectangle detector and a previous dense segmentation trained to detect door regions.

pipeline	**Precision**	**Recall**
Fine-tuned CNN	0.47	0.15
Fine-tuned CNN + [23]	0.43	**0.74**
Superpixel classifier	0.19	0.51
Superpixel classifier + [23]	**0.21**	0.47

These results provide preliminary results on how the segmentation from either a fine-tuned end-to-end CNN model (Fig. 4.2 c) or a superpixel classifier (Fig. 4.2 a) could be improved. When the dense semantic segmentation is very sparse (fine-tuned CNN pipeline), the recall R is very low but the precision P is high, combining the segmentation with the rectangular hypotheses helps to improve the R while the precision is maintained. Besides, when the semantic segmentation seems randoms, the R is high but the P is too

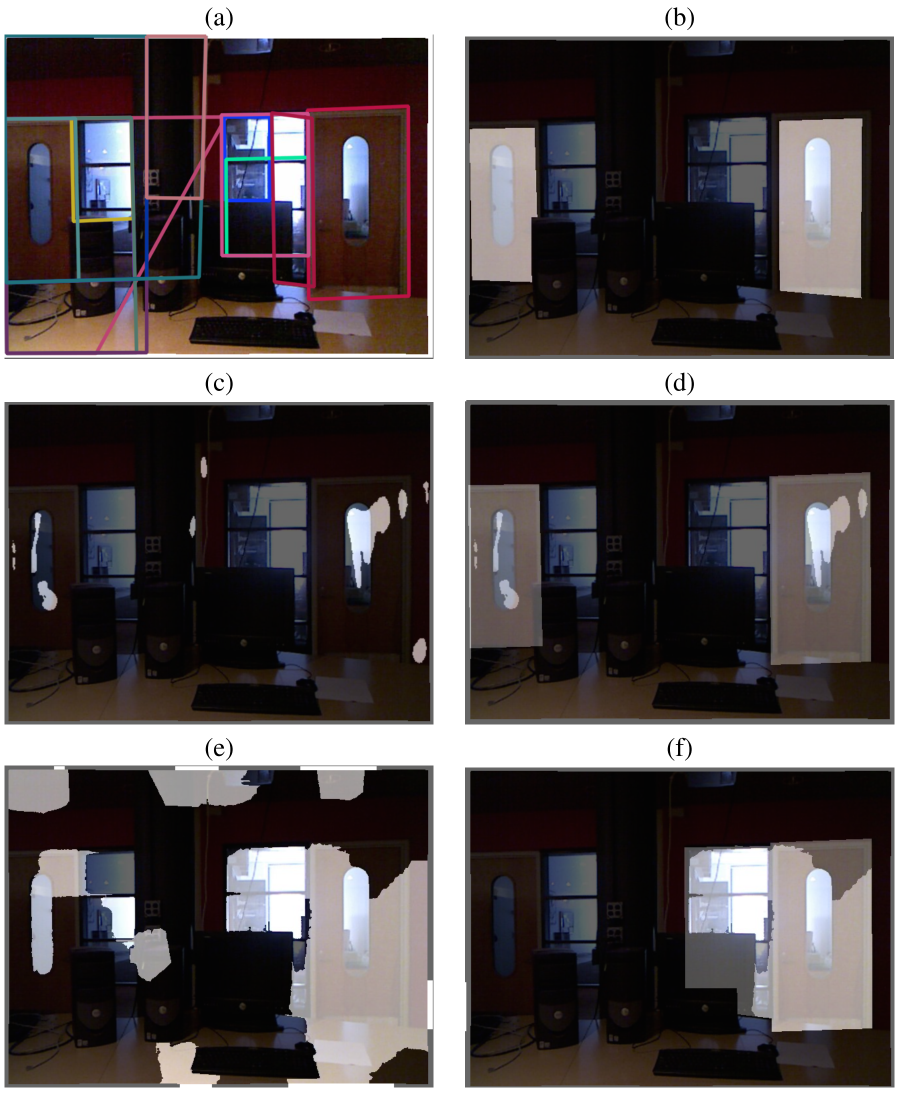


Figure 4.16: Improving Doors detection combining a sign rectangular detector and a dense Door segmentation. (a) Rectangular hypotheses. (b) Doors ground truth. (c) Fine-tunned CNN Door segmentation (pipeline described in Section 4.2.2). (d) Improved previous Door segmentation. (e) Superpixel Door Classifier (pipeline described in Section 4.2.1). (f) Improved previous Door segmentation.

low, using the proposed combination helps to remove the false positive, i.e improve the precision *P* while the *R* is similar.

4.7 Conclusions

Many applications require semantic segmentation of specific classes for which there is not many labeled training data. Recent CNN approaches have proven useful for transfer knowledge from existing models to new tasks. This chapter evaluates the two major CNN-based transfer learning strategies: fine-tuning existing models and CNN based features for classification. Besides, this chapter has proposed how to integrate a superpixel representation with these strategies and combining effectively local and context superpixel description.

The presented three experiments have covered a wide range of semantic segmentation problems. The results with different real scenarios have demonstrated the benefits of the proposed pipelines to learn how to segment new classes with limited data and resources. Although fine-tuning existing model seems the most direct way to transfer information learned on those models, the experiments have shown that using the proposed CNN-based representation and superpixels provides a more effective transfer learning pipeline than fine-tuning a model.

However, there are situations where the segmentation does not provide acceptable results, e.g in the presented Door segmentation. This chapter has presented that easy image processing techniques may be suitable for improving the segmentation results.

Chapter 5

Conclusions and Future Work

This thesis contributes towards vision based systems for automatic understanding of environment and their applications. The general goal of this thesis was to learn complex scene information from pieces of user prior knowledge. In particular, the work on this thesis proposes and demonstrates novel approaches to learn information of interest for specific applications from two different sources of prior information: interactive human input and existing examples of semantically labeled data.

5.1 Dense labeling from interactive user-labeling propagation

Frequently, computer vision algorithms need to operate in interactive applications, such as smart image editing environments. Many image post-processing applications require a dense per-pixel value of certain magnitude, which may be easily initialized in a few pixels taking advantage of user interaction. This is a natural application where the user interactively provides prior knowledge to the system, and the system needs to be able to produce a satisfactory final result requiring as little interaction from the user as possible.

Chapter 2 describes one of the main contributions in this thesis, a newly developed efficient approach to propagate confident but sparse user information to all pixels in an image. The presented implementation has the advantage with respect to previous similar solutions of running at interactive rates. This means our paradigm enables the development of interactive applications that apply any image post processing technique that requires the estimation of continuous magnitudes per pixel.

The interactivity of our approach is guaranteed because it is formulated as a linear system of equations over superpixels. Although our accuracy is limited by the number of superpixels, we have shown that we yield results which are accurate enough for many applications and often comparable to other slower state of the art methods. Besides, since we target an interactive technique, the user can always continue to iteratively refining the input, therefore automatically improving the output, until it achieves the desired level of accuracy.

The interactivity rates of our approach, together with the aforementioned expanded flexibility, provide great potential for image editing applications. To demonstrate the suitability of our approach, we have implemented three interactive applications, detailed in Chapter 3, that apply complex well-known image filters and effects: depth of field, dehazing and tone mapping. These three applications use the same proposed propagation scheme, and their results demonstrate that our generic propagation system achieves comparable image effects than related methods which are ad-hoc for a single specific problem. Besides, thanks to our paradigm, these applications can run as interactive tools, with very low computational requirements. This opens the opportunity to bring this type of editing tools to mobile and embedded devices.

We believe our work can inspire both new interactive editing applications, as well as future research on interactive editing tools. Potential lines of future work that can be inspired from the combination of the presented work and previous work are multi-view approaches for transferring edits between views [65] or for depth editing and navigation [66]. Besides, instead of plain images, our approach could be extended for material appearance [68] or even material transfer [67] applications, given the adequate adaptation and identification of the involved magnitudes to material space.

5.2 Dense labeling from semantic segmentation models

We have seen that semantic image parsing, an important task to the broader scene understanding problem, can be formulated as a dense labeling problem, where the labels corresponds to different concepts depending on the application. Many applications require semantic segmentation of specific classes for which there is not enough labeled data to train a state-of-the-art model from scratch. Recent CNN approaches have proven useful to transfer knowledge from existing models to new tasks.

In this thesis, we have worked with the two major CNN-based transfer learning strategies: fine-tuning existing models and CNN based features for classification. The contributions along this topic include new strategies on how to integrate a superpixel representation with these strategies and combining effectively local and context superpixel description. These strategies are detailed and evaluated in Chapter 4.

The presented experiments to validate the proposed strategies cover a wide range of semantic segmentation problems and application areas. In particular, present how to transfer information from models learned from ground-level images, such as autonomous driving scenarios, to a different image domain, such as aerial images or underwater image data.

These results with data from different real scenarios have demonstrated the benefits of the proposed pipelines to learn how to segment new classes with limited data and resources. Although fine-tuning an existing model seems the most direct way to transfer information learned on those models, the experiments have shown that using the proposed CNN-based representation and superpixels can yield a more effective transfer learning pipeline than fine-tuning a model when resources are limited.

Future works on optimizing and enabling this kind of visual recognition systems on embedded platforms are a challenging problem that would enable autonomous systems to have richer real time perception modules.

Our experimentation also shows use cases where the proposed strategies to learn new classes semantic segmentation are not sophisticated enough for the targeted problem, and therefore does not provide acceptable results. In this respect, this thesis introduces two additional contributions that are complementary to the previous approach, and present promising improvements in the segmentation results of these more complex use cases.

For example, in the presented *Door* class segmentation problem, the target class door has too large variability of appearances for the limited amount of data used for training. In this kind of problems, where appearance is very heterogeneous and the object is often strongly occluded, we have presented how simple geometry based image processing techniques improve the segmentation results.

On other use cases, the problem encountered is that the available labels for training are not accurate or dense enough to train the desired model. In our case of study, available sparse labels on underwater imagery are not enough to train a common dense labeling model to detect coral regions in those images. In this case, we have presented a simple strategy to augment incomplete input data. Our results show that although this augmentation is approximated and somehow noisy, it still yields better dense segmentation models than prior work on the same data.

The promising results obtained, together with recent related work exploring related ideas, encourage future lines of work along similar strategies, i.e., to refine either the input data or the output results of deep learning based models. In particular, next steps to improve these systems will consider how to deal with incomplete or noisy image labels for training, and strategies to combine existing geometric information obtained from well-known approaches to refine less precise predictions of generic models.

Part I

Appendix

Appendix A

Additional results of our interactive editing applications

As previously described in Chapter 2, one of the main contributions of this thesis is an interactive paradigm for label propagation from user interactions. This Appendix includes additional results for the three presented real use cases applying this paradigm: tone mapping, depth of field and dehazing effects simulation, as detailed in Chapter 3.

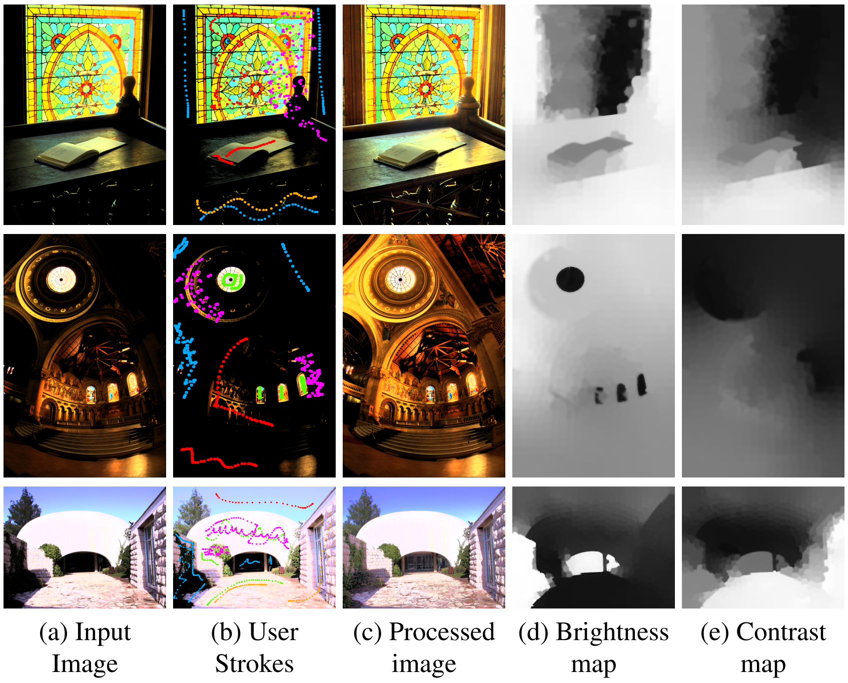


Figure A.1: Additional examples of our interactive HDR tone mapping application where an input HDR image is converted to a LDR output image. Given an input image (a), the user provides a few strokes (b) to adjust the brightness and contrast in different regions in the final processed image (c). The estimated brightness (d) and contrast (e) maps are interactively updated after each user edition and are used for generating the processed image. Blue, green and red strokes mark regions need to be darken, illuminated and unmodified respectively; while pink and orange strokes mark regions where reduce or increase the contrast respectively.

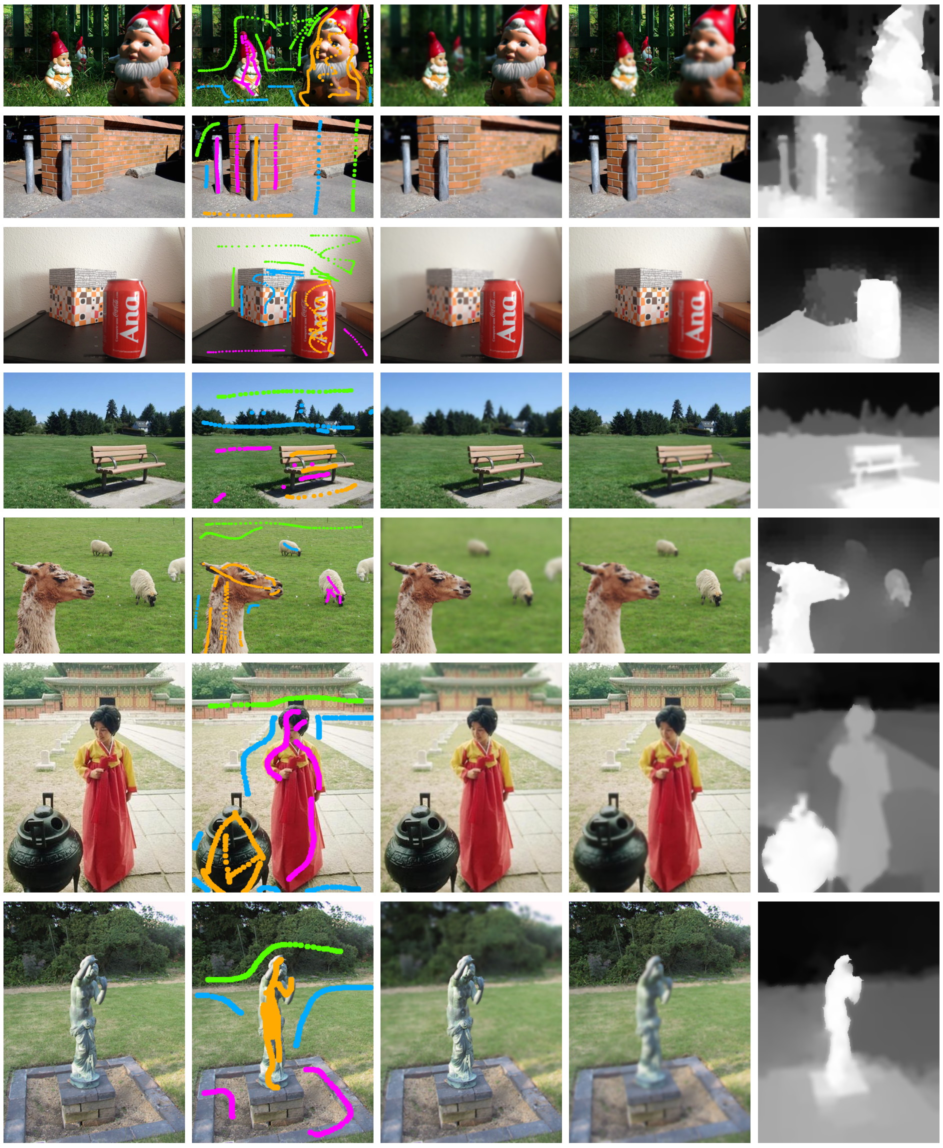


Figure A.2: Additional examples of our interactive depth of field application. Given an input image (a), the user provides a few strokes (b) to add depth of field effects in the processed image (c). The user also can change the focus point . The estimated depth map (d) is interactively updated after each user edition and is used for generating a depth of field effect. Orange strokes mark foreground objects; pink, blue and green strokes mark background objects at different depths.

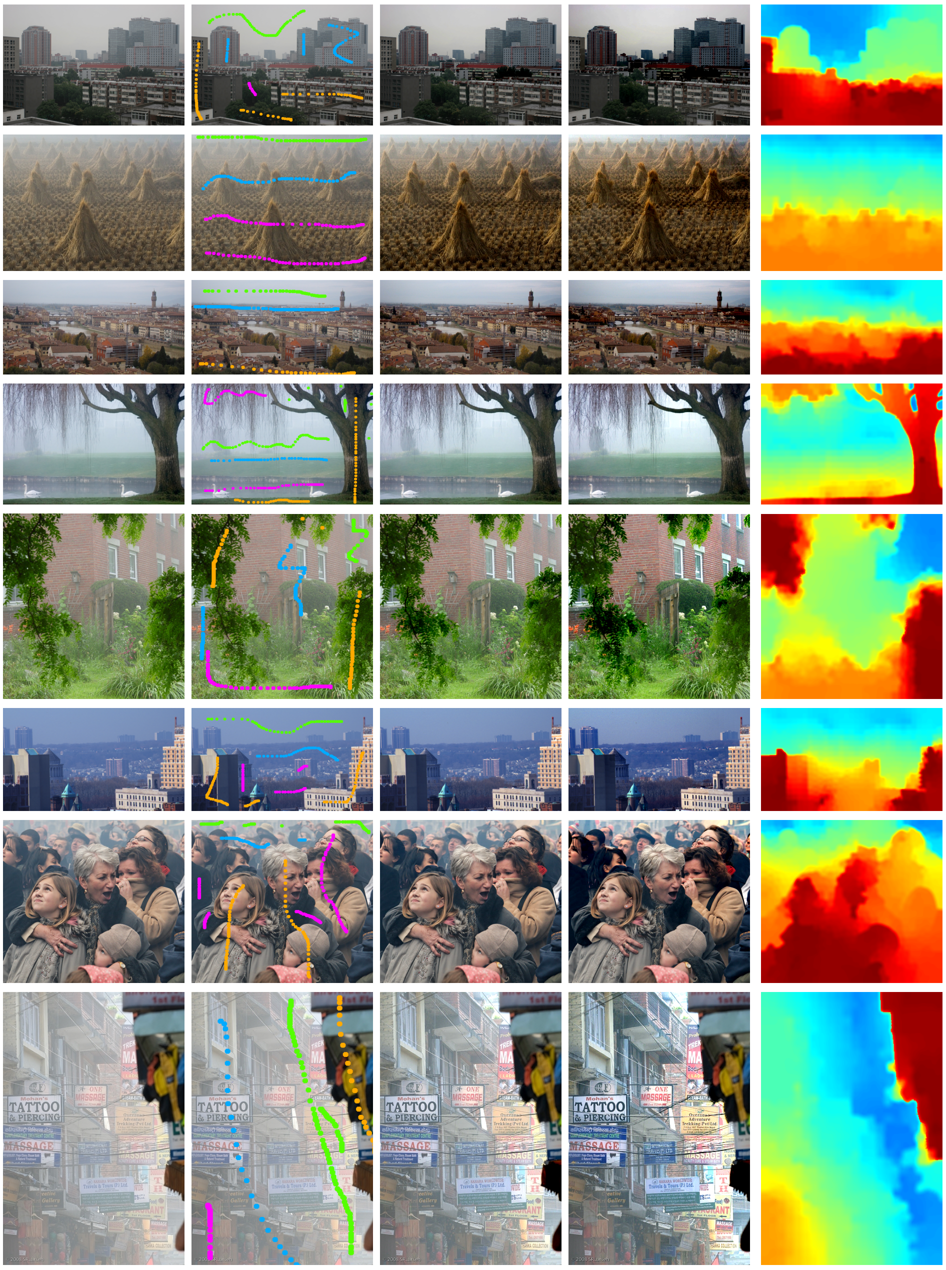

(a) Input Image (b) User Strokes (c) Processed images (d) Transmittance estimation

Figure A.3: Additional examples of our interactive dehazing application. Given an input image (a), the user provides a few strokes (b) to remove the fog effects in the processed image (c). The transmittance map (d) is interactively updated after each user edition and is used for applying the dehazing effect. Orange strokes mark regions without fog; pink, blue and green strokes mark regions with different levels of fog.

Appendix B

Additional results of transfer learning pipelines.

As previously described in Chapter 4, the second part of this thesis investigates how to improve systems that learn dense semantic labeling of images from user labeled examples. This Appendix includes additional results for the three presented region segmentation (Text, Door and Water), obtained with two major CNN-based strategies for transfer learning.

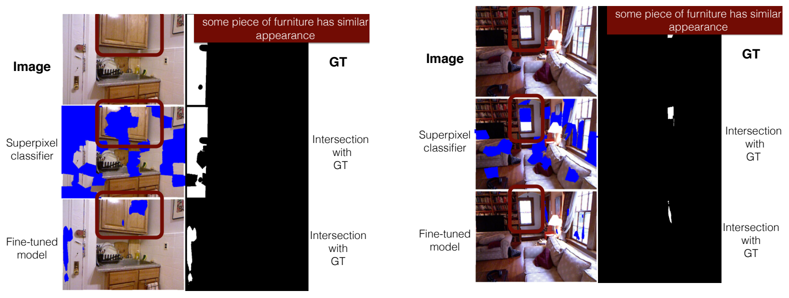


Figure B.1: Doors region segmentation results. It is a more challenging problem, as we can see, there are objects with similar appearance.

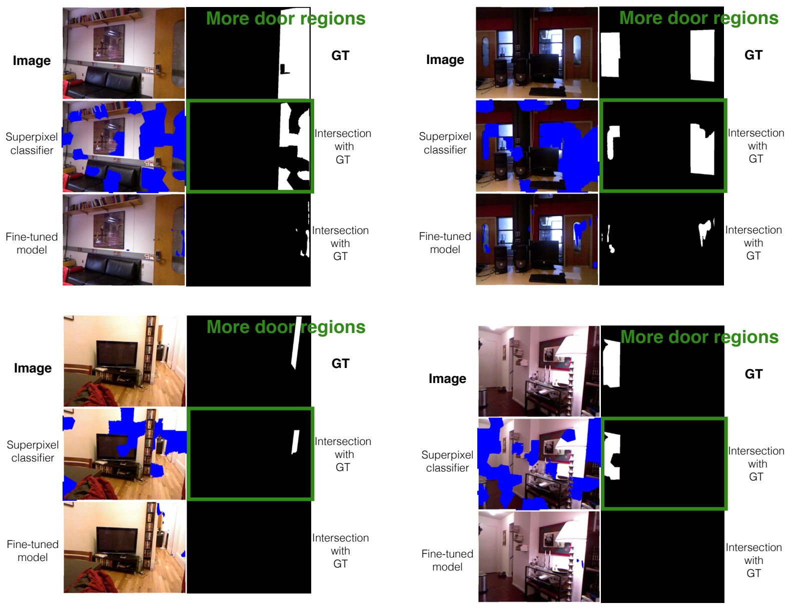


Figure B.2: Doors region segmentation results. Superpixel classifier detects more doors region than Fine-tuned pipeline.

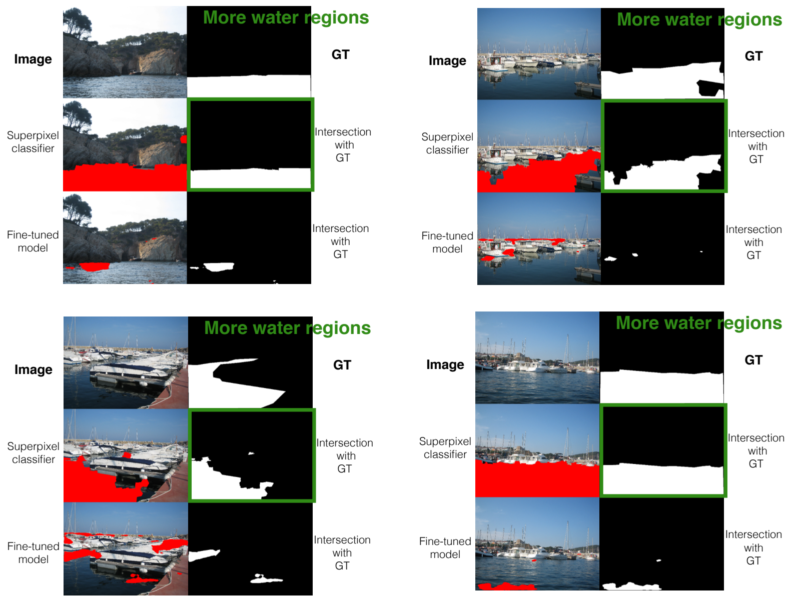


Figure B.3: Water region segmentation results. Superpixel classifier detects more water region than Fine-tuned pipeline.

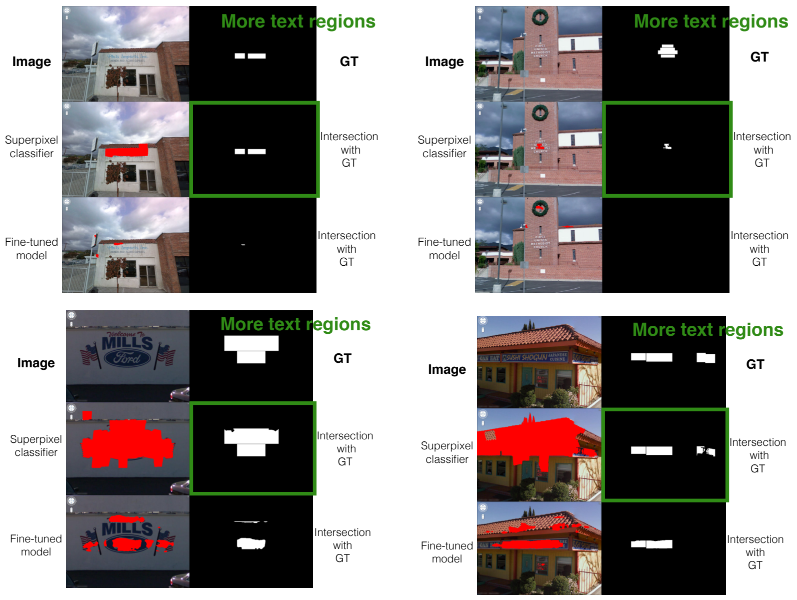


Figure B.4: Text region segmentation results. Some example where superpixel classifier detects more text regions than fine-tuned model.

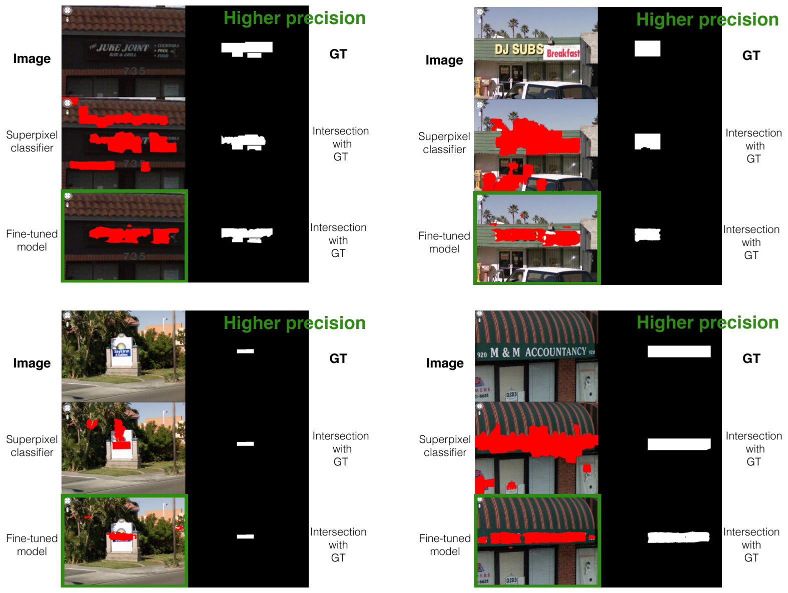


Figure B.5: Text region segmentation results. Fine-tuned pipeline detects text regions with higher precision.

Appendix C

Additional results of our post-proccesing segmentation.

This Appendix includes additional results of Door segmentation problem. As we mentioned before, we propose to add a light geometry based post-processing step, in order to improve the segmentation. We include intermediate segmentation results, in order to describe how the rectangular regions are combined with the original segmentation.

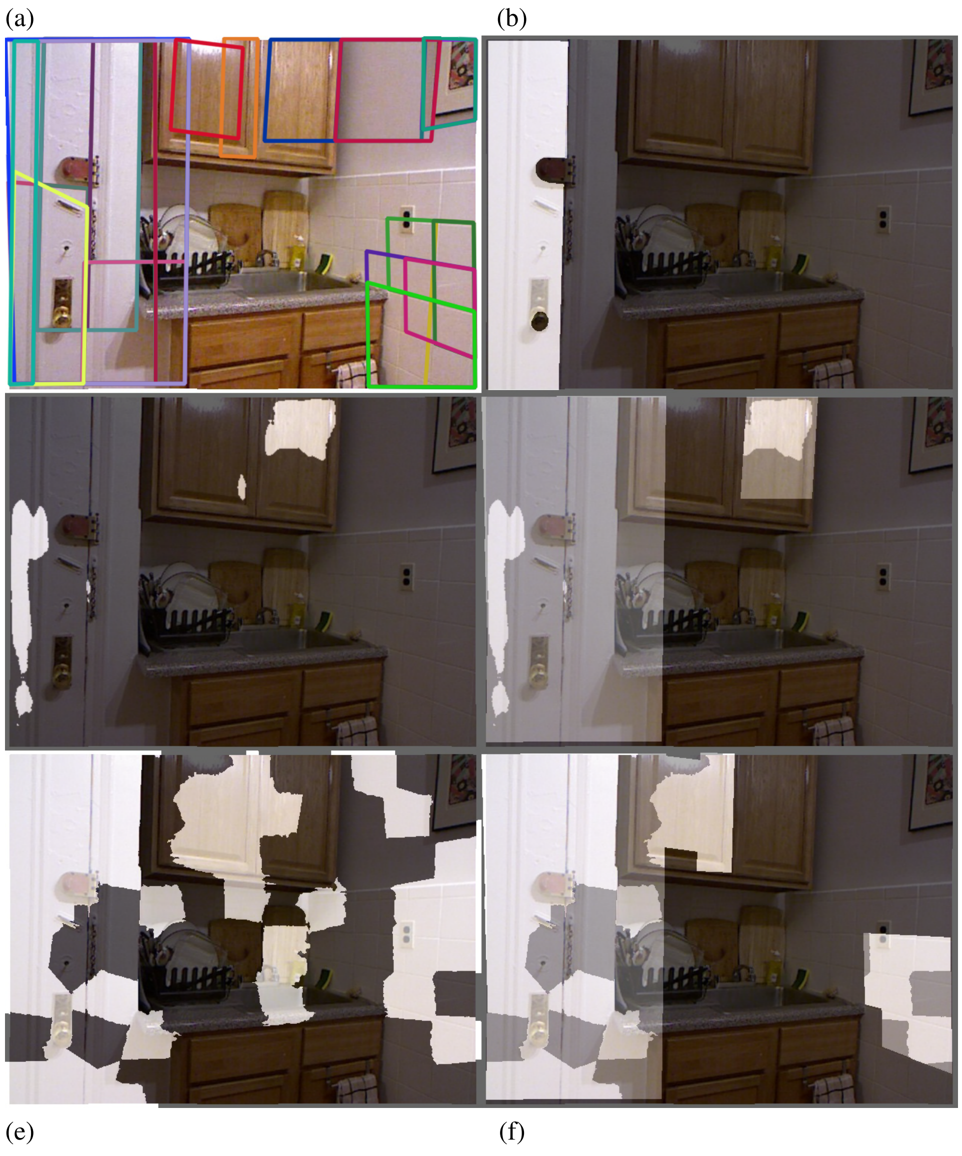
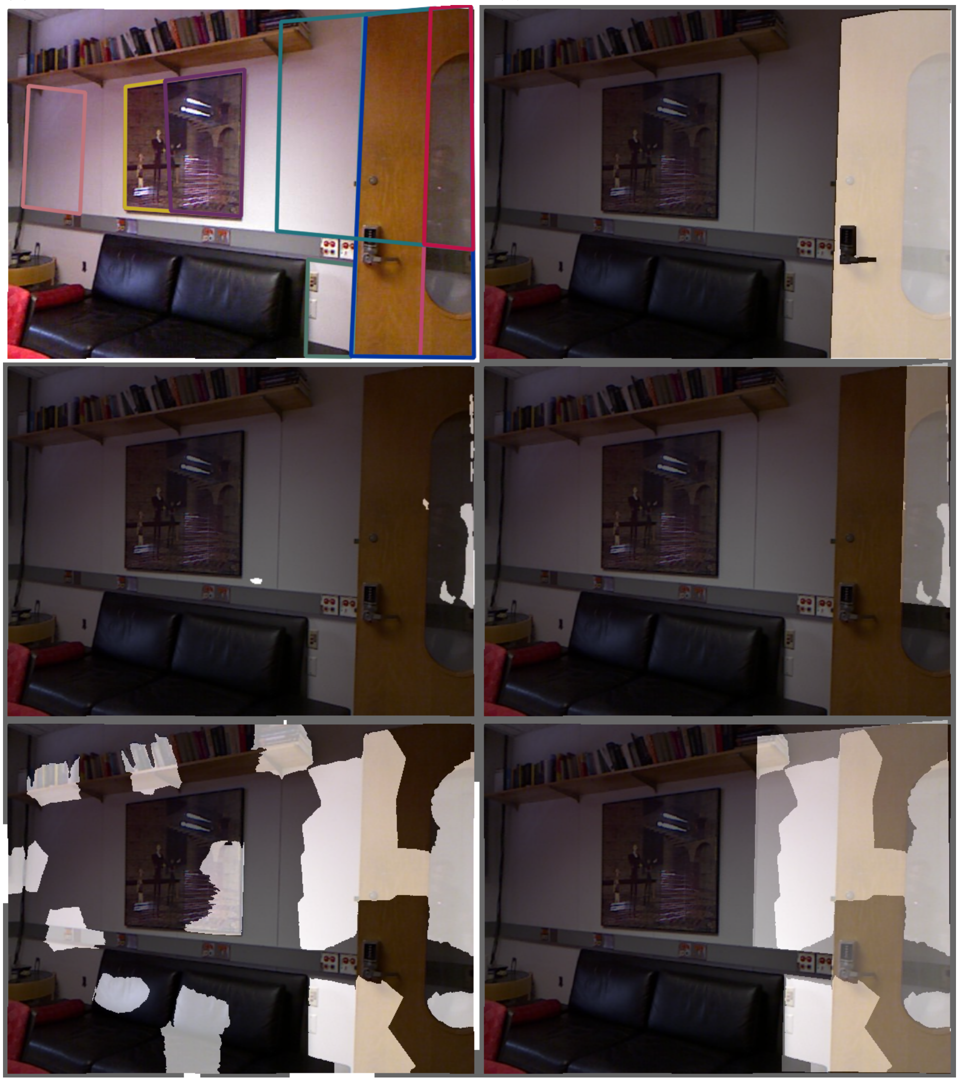


Figure C.1: Improving Doors detection combining a sign rectangular detector and a dense Door segmentation. (a) Rectangular hypotheses. (b) Doors ground truth. (c) Fine-tunned CNN Door segmentation (pipeline described in Section 4.2.2). (d) Improved previous Door segmentation. (e) Superpixel Door Classifier (pipeline described in Section 4.2.1). (f) Improved previous Door segmentation.

(a) (b)



(e) (f)

Figure C.2: Improving Doors detection combining a sign rectangular detector and a dense Door segmentation. (a) Rectangular hypotheses. (b) Doors ground truth. (c) Fine-tunned CNN Door segmentation (pipeline described in Section 4.2.2). (d) Improved previous Door segmentation. (e) Superpixel Door Classifier (pipeline described in Section 4.2.1). (f) Improved previous Door segmentation.

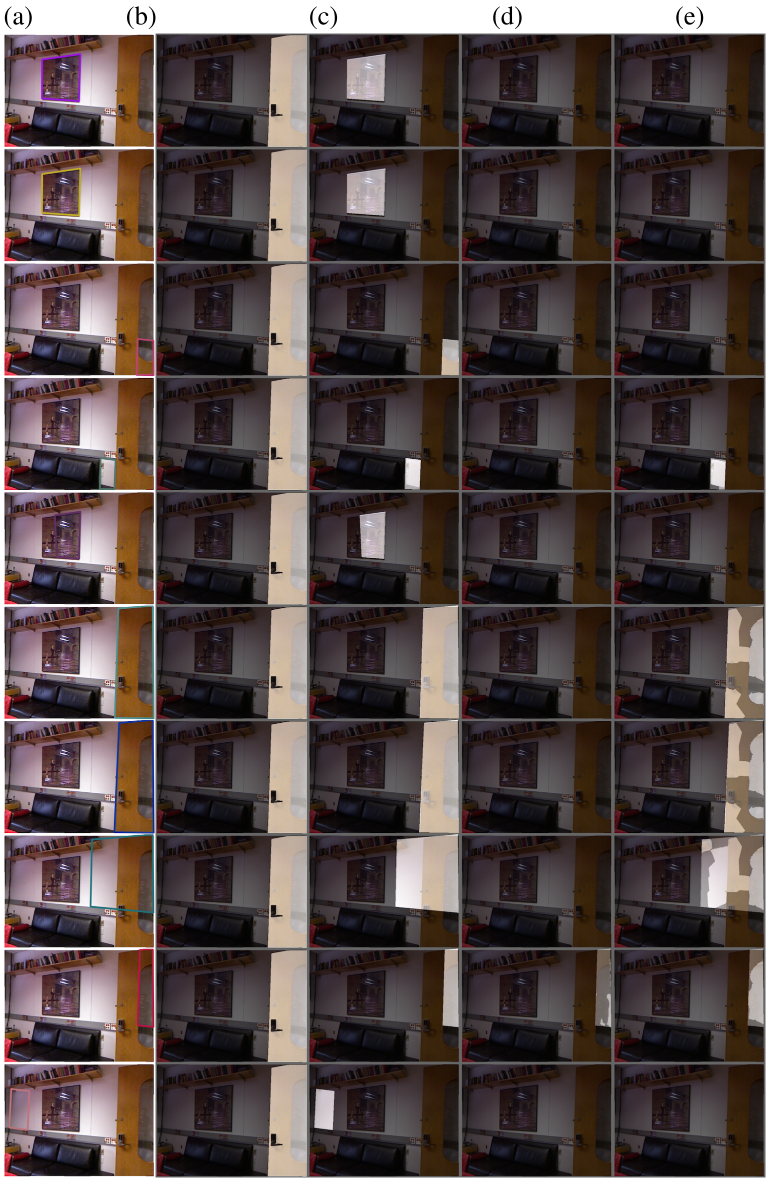


Figure C.3: Intermediate step to improve Doors detection combining a sign rectangular detector and a dense Door segmentation. (a) A rectangular hypothesis. (b) Doors ground truth. (c) The rectangular segmentation hypothesis. (d) Intersection between Fine-tunned CNN Door segmentation and the rectangular hypothesis. (e) Intersection between Superpixel Door Classifier and the rectagular segmentation hyphothesis.

Bibliography

[1] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “Deepface: Closing the gap to human-level performance in face verification,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[2] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik, “Indoor scene understanding with rgb-d images: Bottom-up segmentation, object detection and semantic segmentation,” *International Journal of Computer Vision*, vol. 112, no. 2, pp. 133–149, 2015.

[3] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, “Social lstm: Human trajectory prediction in crowded spaces,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 961–971, 2016.

[4] H. S. Parekh, D. G. Thakore, and U. K. Jaliya, “A survey on object detection and tracking methods,” *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 2, no. 2, pp. 2970–2979, 2014.

[5] P. Kamavisdar, S. Saluja, and S. Agrawal, “A survey on image classification approaches and techniques,” *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, no. 1, pp. 1005–1009, 2013.

[6] J. Deng, J. Krause, and L. Fei-Fei, “Fine-grained crowdsourcing for fine-grained recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2013.

[7] O. Russakovsky, L.-J. Li, and L. Fei-Fei, “Best of both worlds: human-machine collaboration for object annotation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2121–2131, 2015.

[8] H. Wang, S. Gong, X. Zhu, and T. Xiang, “Human-in-the-loop person re-identification,” in *European Conference on Computer Vision*, pp. 405–422, Springer, 2016.

[9] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *arXiv preprint arXiv:1511.00561*, 2015.

[10] N. Sharma and L. M. Aggarwal, "Automated medical image segmentation techniques," *Journal of medical physics/Association of Medical Physicists of India*, vol. 35, no. 1, p. 3, 2010.

[11] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," in *ACM Transactions on Graphics*, vol. 23, pp. 689–694, ACM, 2004.

[12] H. Ao, Y. Zhang, A. Jarabo, B. Masia, Y. Liu, D. Gutierrez, and Q. Dai, "Light field editing based on reparameterization," in *Pacific Rim Conference on Multimedia*, 2015.

[13] A. Bousseau, S. Paris, and F. Durand, "User assisted intrinsic images," *ACM Transactions on Graphics (SIGGRAPH Asia 2009)*, vol. 28, no. 5, 2009.

[14] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, pp. 3431–3440, 2015.

[15] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pp. 1175–1183, IEEE, 2017.

[16] L. Schneider, M. Cordts, T. Rehfeld, D. Pfeiffer, M. Enzweiler, U. Franke, M. Pollefeys, and S. Roth, "Semantic stixels: Depth is not enough," in *Intelligent Vehicles Symposium*, pp. 110–117, IEEE, 2016.

[17] D. Kochanov, A. Ošep, J. Stückler, and B. Leibe, "Scene flow propagation for semantic mapping and object discovery in dynamic street scenes," in *IROS*, pp. 1785–1792, IEEE, 2016.

[18] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Conf. on CVPR*, pp. 580–587, 2014.

[19] L. Ballan, F. Castaldo, A. Alahi, F. Palmieri, and S. Savarese, "Knowledge transfer for scene-specific motion prediction," in *European Conference on Computer Vision*, pp. 697–713, Springer, 2016.

[20] A. B. Cambra, A. C. Murillo, and A. Muñoz, "A generic tool for interactive complex image editing," *The Visual Computer*, Aug 2017.

[21] A. B. Cambra, A. Muñoz, J. J. Guerrero, and A. C. Murillo, "Dense labeling with user interaction: an example for depth-of-field simulation," in *British Machine Vision Conference*, September 2016.

[22] A. B. Cambra, A. Muñoz, A. Murillo, J. Guerrero, and D. Gutierrez, "Improving depth estimation using superpixels," in *Spanish Computer Graphics Conf.*, pp. 49–58, The Eurographics Assoc., 2014.

[23] A. B. Cambra and A. Murillo, "Towards robust and efficient text sign reading from a mobile phone," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 64–71, IEEE, 2011.

[24] A. B. Cambra, A. Muñoz, and A. C. Murillo, "How to transfer an autonomous driving model for semantic segmentation to other domains?," in *Third Iberian Robotics Conference*, 2017.

[25] I. Alonso, A. B. Cambra, T. Muñoz, Adolfo andTreibitz, and A. Murillo, "Coral-segmentation: Training dense labeling models with sparse ground truth. towards robust and efficient text sign reading from a mobile phone," in *Computer Vision Workshops (ICCV Workshops), 2017 IEEE International Conference on*, IEEE, 2017.

[26] B. Mičušík and J. Košecká, "Multi-view superpixel stereo in urban environments," *International Journal of Computer Vision*, vol. 89, no. 1, pp. 106–119, 2010.

[27] X. Ren, L. Bo, and D. Fox, "Rgb-(d) scene labeling: Features and algorithms," in *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 2759–2766, IEEE, 2012.

[28] J. Tighe and S. Lazebnik, "Superparsing," *International Journal of Computer Vision*, vol. 101, no. 2, pp. 329–349, 2013.

[29] Q. Luan, F. Wen, D. Cohen-Or, L. Liang, Y.-Q. Xu, and H.-Y. Shum, "Natural image colorization," in *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pp. 309–320, Eurographics Association, 2007.

[30] S. Bergmann, T. Ritschel, and C. Dachsbacher, "Interactive appearance editing in RGB-D images," in *19th International Workshop on Vision, Modeling and Visualization*, pp. 1–8, Eurographics Association, 2014.

[31] F. Di Renzo, C. Calabrese, and F. Pellacini, "Appim: linear spaces for image-based appearance editing," *ACM Transactions on Graphics*, vol. 33, no. 6, p. 194, 2014.

[32] N. Bonneel, K. Sunkavalli, J. Tompkin, D. Sun, S. Paris, and H. Pfister, "Interactive intrinsic video editing," *ACM Transactions on Graphics*, vol. 33, no. 6, p. 197, 2014.

[33] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for markov random fields with smoothness-based priors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 6, pp. 1068–1080, 2008.

[34] Q. Chen and V. Koltun, “Fast MRF optimization with application to depth reconstruction,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 3914–3921, 2014.

[35] R. Shen, I. Cheng, X. Li, and A. Basu, “Stereo matching using random walks,” in *International Conference on Pattern Recognition*, pp. 1–4, 2008.

[36] O. Teboul, L. Simon, P. Koutsourakis, and N. Paragios, “Segmentation of building facades using procedural shape priors,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 3105–3112, IEEE, 2010.

[37] A. Fabijanska and J. Goclawski, “The segmentation of 3d images using the random walking technique on a randomly created image adjacency graph.,” *IEEE Transactions Image Processing*, vol. 24, no. 2, p. 524, 2015.

[38] L. Grady, “Random walks for image segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1768–1783, 2006.

[39] S. Andrews, G. Hamarneh, and A. Saad, “Fast random walker with priors using precomputation for interactive medical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention*, pp. 9–16, Springer, 2010.

[40] D. Freedman, “An improved image graph for semi-automatic segmentation,” *Signal, Image and Video Processing*, vol. 6, no. 4, pp. 533–545, 2012.

[41] C. Wah, S. Branson, P. Perona, and S. Belongie, “Multiclass recognition and part localization with humans in the loop,” in *IEEE Int. Conf. on Computer Vision*, pp. 2524–2531, Nov 2011.

[42] C.-H. Lu, Y.-C. Ho, Y.-H. Chen, and L.-C. Fu, “Hybrid user-assisted incremental model adaptation for activity recognition in a dynamic smart-home environment,” *IEEE Transactions on Human-Machine Systems*, vol. 43, pp. 421–436, Sept 2013.

[43] O. Razeghi, G. Qiu, H. Williams, K. Thomas, and I. VIPLAB, “Skin lesion image recognition with computer vision and human in the loop,” *Medical Image Understanding and Analysis (MIUA)*, pp. 167–172, 2012.

[44] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, “SLIC superpixels compared to state-of-the-art superpixel methods,” *IEEE Trans. on PAMI*, vol. 34, no. 11, pp. 2274–2282, 2012.

[45] D. Scharstein and R. Szeliski, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[46] D. Scharstein and R. Szeliski, “High-accuracy stereo depth maps using structured light,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 1, pp. I–195, 2003.

[47] J. Besag, “On the statistical analysis of dirty pictures,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 259–302, 1986.

[48] Y. Boykov, O. Veksler, and R. Zabih, “Fast approximate energy minimization via graph cuts,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.

[49] M. F. Tappen and W. T. Freeman, “Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters,” in *IEEE International Conference on Computer Vision*, pp. 900–906, IEEE, 2003.

[50] V. Kolmogorov, “Convergent tree-reweighted message passing for energy minimization,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1568–1583, 2006.

[51] D. Weinshall, “Qualitative depth from stereo, with applications,” in *Computer Vision, Graphics, and Image Processing*, pp. 222–241, 1990.

[52] F. Yu and D. Gallup, “3d reconstruction from accidental motion,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 3986–3993, 2014.

[53] M. Liu, M. Salzmann, and X. He, “Discrete-continuous depth estimation from a single image,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 716–723, IEEE, 2014.

[54] R. Fattal, “Dehazing using color-lines,” *ACM Transaction on Graphics*, vol. 34, no. 13, 2014.

[55] A. Jarabo, B. Masia, A. Bousseau, F. Pellacini, and D. Gutierrez, “How do people edit light fields?,” *ACM Transactions on Graphics (SIGGRAPH 2014)*, vol. 33, no. 4, 2014.

[56] M. Kraus and M. Strengert, “Depth-of-field rendering by pyramidal image processing,” *Computer Graphics Forum*, vol. 26, no. 3, pp. 645–654, 2007.

[57] D. Berman, T. Treibitz, and S. Avidan, “Non-local image dehazing,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 1674–1682, June 2016.

[58] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda, “Photographic tone reproduction for digital images,” *ACM Transactions on Graphics*, vol. 21, pp. 267–276, July 2002.

[59] R. Mantiuk and H.-P. Seidel, “Modeling a generic tone-mapping operator,” in *Computer Graphics Forum*, vol. 27, pp. 699–708, Wiley Online Library, 2008.

[60] X. Chen, D. Zou, Q. Zhao, and P. Tan, “Manifold preserving edit propagation,” *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, p. 132, 2012.

[61] J. Ning, L. Zhang, D. Zhang, and C. Wu, "Interactive image segmentation by maximal similarity based region merging," *Pattern Recognition*, vol. 43, no. 2, pp. 445–456, 2010.

[62] A. Lopez, E. Garces, and D. Gutierrez, "Depth from a Single Image Through User Interaction," in *Spanish Computer Graphics Conf.*, pp. 11–20, The Eurographics Assoc., 2014.

[63] S. Iizuka, Y. Endo, Y. Kanamori, J. Mitani, and Y. Fukui, "Efficient depth propagation for constructing a layered depth image from a single image," *Computer Graphics Forum (Proc. of Pacific Graphics 2014)*, vol. 33, no. 7, pp. 279–288, 2014.

[64] K. Yücer, A. Sorkine-Hornung, and O. Sorkine-Hornung, "Transfusive weights for content-aware image manipulation," in *Proceedings of the Symposium on Vision, Modeling and Visualization (VMV)*, Eurographics Association, 2013.

[65] K. Yücer, A. Jacobson, A. Hornung, and O. Sorkine, "Transfusive image manipulation," *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA)*, vol. 31, no. 6, pp. 176:1–176:9, 2012.

[66] G. Chaurasia, S. Duchene, O. Sorkine-Hornung, and G. Drettakis, "Depth synthesis and local warps for plausible image-based navigation," *ACM Transactions on Graphics*, vol. 32, no. 3, p. 30, 2013.

[67] X. An, X. Tong, J. D. Denning, and F. Pellacini, "AppWarp: Retargeting measured materials by appearance-space warping," *ACM Trans. Graph.*, vol. 30, pp. 147:1–147:10, Dec. 2011.

[68] F. Di Renzo, C. Calabrese, and F. Pellacini, "AppIm: Linear spaces for image-based appearance editing," *ACM Trans. Graph.*, vol. 33, pp. 194:1–194:9, Nov. 2014.

[69] D. Lischinski, Z. Farbman, M. Uyttendaele, and R. Szeliski, "Interactive local adjustment of tonal values," *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3, pp. 646–653, 2006.

[70] O. Wang, M. Lang, M. Frei, A. Hornung, A. Smolic, and M. Gross, "StereoBrush: interactive 2d to 3d conversion using discontinuous warps," in *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pp. 47–54, ACM, 2011.

[71] X. An and F. Pellacini, "AppProp: all-pairs appearance-space edit propagation," in *ACM Transactions on Graphics (TOG)*, vol. 27, p. 40, ACM, 2008.

[72] K. Xu, Y. Li, T. Ju, S.-M. Hu, and T.-Q. Liu, "Efficient affinity-based edit propagation using k-d tree," *ACM Trans. Graph.*, vol. 28, pp. 118:1–118:6, Dec. 2009.

[73] C. Liang-Chieh, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," in *International Conference on Learning Representations*, 2015.

[74] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *IEEE International Conf. on Computer Vision*, 2015.

[75] G.-J. Qi, "Hierarchically gated deep networks for semantic segmentation," in *CVPR*, pp. 2267–2275, 2016.

[76] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *CVPR*, pp. 4340–4349, 2016.

[77] P. O. Pinheiro, R. Collobert, and P. Dollar, "Learning to segment object candidates," in *NIPS*, pp. 1990–1998, 2015.

[78] S. Bell, C. Lawrence Zitnick, K. Bala, and R. Girshick, "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks," in *CVPR*, pp. 2874–2883, 2016.

[79] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *CVPR*, pp. 2414–2423, 2016.

[80] H. C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," *IEEE T. on Medical Imaging*, vol. PP, no. 99, pp. 1–1, 2016.

[81] J. Tighe and S. Lazebnik, "Superparsing: Scalable nonparametric image parsing with superpixels," *IJCV*, vol. 101, no. 2, pp. 329–349, 2013.

[82] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. of the 31st Int. Conf. on Machine Learning*, pp. 647–655, 2014.

[83] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, pp. 1097–1105, 2012.

[84] G. Chéron, I. Laptev, and C. Schmid, "P-cnn: Pose-based cnn features for action recognition," in *Proc. of the IEEE International Conf. on Computer Vision*, pp. 3218–3226, 2015.

[85] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *IEEE Conf. on CVPR*, pp. 815–823, 2015.

[86] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *NIPS*, 2015.

[87] K. Sakurada and T. Okatani, “Change detection from a street image pair using cnn features and superpixel segmentation,” *British Machine Vision Conference (BMVC)*, 2015.

[88] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich, “Feedforward semantic segmentation with zoom-out features,” in *Proc. of the IEEE Conf. on CVPR*, pp. 3376–3385, 2015.

[89] C. L. Zitnick and P. Dollár, “Edge boxes: Locating object proposals from edges,” in *European Conf. on Computer Vision*, 2014.

[90] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *CVPR*, pp. 886–893, IEEE, 2005.

[91] S. Dutt Jain and K. Grauman, “Active image segmentation propagation,” in *Proc. of the IEEE Conf. on CVPR*, pp. 2864–2873, 2016.

[92] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, “Learning deep features for scene recognition using places database,” in *NIPS*, pp. 487–495, 2014.

[93] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2014.

[94] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “A training algorithm for optimal margin classifiers,” in *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 144–152, ACM, 1992.

[95] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *CVPR*, pp. 248–255, IEEE, 2009.

[96] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, “Densely connected convolutional networks,” *arXiv preprint arXiv:1608.06993*, 2016.

[97] K. Wang and S. Belongie, “Word spotting in the wild,” in *ECCV*, pp. 591–604, 2010.

[98] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, “Indoor segmentation and support inference from rgbd images,” in *ECCV*, 2012.

[99] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, “Labelme: a database and web-based tool for image annotation,” *IJCV*, vol. 77, no. 1-3, pp. 157–173, 2008.

[100] C.-C. Chang and C.-J. Lin, “LIBSVM: a library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[101] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in *Proc. of the 22Nd ACM International Conference on Multimedia*, MM ’14, pp. 675–678, ACM, 2014.

[102] A. N. Venkitasubramanian, T. Tuytelaars, and M.-F. Moens, “Wildlife recognition in nature documentaries with weak supervision from subtitles and external data,” *Pattern Recogn. Lett.*, vol. 81, pp. 63–70, Oct. 2016.

[103] O. Beijbom, T. Treibitz, D. I. Kline, G. Eyal, A. Khen, B. Neal, Y. Loya, B. G. Mitchell, and D. Kriegman, “Improving automated annotation of benthic survey images using wide-band fluorescence,” *Scientific reports*, vol. 6, 2016.

[104] M. Van den Bergh, X. Boix, G. Roig, B. de Capitani, and L. Van Gool, “Seeds: Superpixels extracted via energy-driven sampling,” in *Computer Vision–ECCV 2012*, pp. 13–26, Springer, 2012.